Analyse der softwarearchitektonischen Unterstützung der Usability mobiler Anwendungen

Dissertation

zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften
(Dr. rer. nat.)

durch die Fakultät für Wirtschaftswissenschaften der
Universität Duisburg-Essen
Campus Essen

vorgelegt von
Bettina Biel, M.A.
geboren in Plauen
Essen (2017)

Tag der mündlichen Prüfung: 6. April 2017

Erstgutachter: Prof. Dr. Volker Gruhn
Zweitgutachter: Prof. Dr. Michael Goedicke

Bettina Biel: *Analyse der softwarearchitektonischen Unterstützung der Usability mobiler Anwendungen*, 2017

ZUSAMMENFASSUNG

Rasante Innovationszyklen, kurze Produkteinführungszeiten und ein hoher Konkurrenzdruck sind typische Rahmenbedingungen für die Entwicklung mobiler Anwendungen. Dies sind Anwendungen, die auf mobilen Endgeräten laufen und in verschiedenen Umgebungen verwendet werden. Usability (Benutzbarkeit) kann durch die Softwarearchitektur einer Anwendung unterstützt, aber auch behindert werden. Je später im Softwareentwicklungsprozess Usability beachtet wird, umso aufwendiger werden Änderungen an der Softwarearchitektur. Um dieses Risiko zu verringern, ist es nötig, so früh wie möglich offenzulegen, ob Usability-Anforderungen architektonisch unterstützt werden.

Potenziell hohe Architekturänderungen werden mit Methoden zur szenariobasierten Softwarearchitekturanalyse ermittelt; es wird dabei verifiziert, dass ein Qualitätsmerkmal architektonisch berücksichtigt wurde. Frühere Methoden bezüglich Usability erreichen dieses Ziel, erscheinen aber sehr komplex: Sie erfordern zum einen Wissen über Patterns und zum anderen sind die Freiheitsgrade beim Erstellen, Auswählen und Evaluieren von Szenarios hoch. Wie die früheren Methoden Usability-Attribute verwenden, behindert zudem eine engere Kooperation mit dem Usability Engineering, für das gemeinsame Begriffe und Vorgehensweisen grundlegend wären. Deshalb ist es notwendig, eine Methode zu konstruieren, die einfacher und interdisziplinärer ausgerichtet ist.

Aufgrund dessen werden in dieser Forschungsarbeit mittels Literaturstudien zuerst Forschungsfragen, dann Hilfsmittel und schließlich eine theoretisch fundierte Methode erarbeitet. Um diese zu validieren und zu vereinfachen, durchläuft sie – mit kanonischer anwendungsnaher Forschung – zwei Fallstudien zu mobilen Anwendungen.

Ergebnis ist die szenario-basierte Methode SATURN („SoftwareArchitekTuranalyse von Usability-anfoRderungeN“), in der anfangs mittels Nutzungskontextanalyse die Interaktionsszenarios erstellt werden, die für Anwender relevant sind. Hilfsmittel umfassen die Faktoren des mobilen Nutzungskontexts und einen Katalog von 50 potenziell architektursensitiven Interaktionsszenarios. Diese sind von Patterns abgeleitet, referenzieren sie und unterliegen einem definierten Lebenszyklus. Die Analyse stützt sich auf die verwendete Architekturdefinition und auf das Prinzip der Sichtenmodelle. Bewertet wird, inwiefern Struktur oder Verhalten von architektonischen Elementen verhindern können, dass ein Interaktionsszenario (hypothetisch) durchgeführt werden kann. Betrachtet wird dabei, wie Usability berücksichtigt wurde, welche vor- und nachteiligen Architekturentscheidungen und welche Austauschbeziehungen mit anderen Qualitätsmerkmalen bestehen. Die Ergebnisse von SATURN fließen zurück zur Erstellung der Softwarearchitektur und zum Usability Engineering. Die Methode ist auch mit einem Nutzertest kombinierbar.

Mit SATURN ist die Analyse der architektonischen Unterstützung für die Usability mobiler Anwendungen einfacher als mit früheren Arbeiten. Dies inspiriert zu weiterer Forschung, wie beispielsweise Fallstudien zum Zusammenhang zwischen Usability und Softwarearchitektur, die Ausrichtung der Methode auf andere Qualitätsmerkmale, neue konstruktive Möglichkeiten in agilen Prozessen oder allgemein die Koordination von Usability Engineering und Softwareentwicklung.

ABSTRACT

The software development of mobile applications, i.e. applications which run on mobile devices and are used in various environments, faces a fast time to market, high competitive pressure, short technical innovation cycles, and high user expectations regarding usability. The software architecture of an application can support but also constrain usability. The later in software development usability is considered, the costlier architectural modifications become. In order to reduce this risk, it is necessary to discover usability requirements that are not supported architecturally as early as possible.

Potentially high architectural changes are elicited using scenario-based software architecture analyses; it is verified, that a quality factor was considered architecturally. Earlier works regarding usability achieve this, though they appear very complex: they depend on the knowledge of patterns and leave room for interpretation while creating, selecting and evaluating scenarios. Also, respective uses of usability attributes hinder a further cooperation with usability engineering, because common terms and methods are a prerequisite for this. Therefore, it is necessary to construct a new method which is easier to use and more interdisciplinary-oriented.

In order to describe research questions, means for the method and a first version of the method itself, literature studies are conducted. Afterwards, the method is validated and improved through two case studies with mobile applications adhering to canonical action research.

In the new scenario-based method SATURN („Software ArchitecTure analysis of Usability-RequiremeNts“), interaction scenarios that are relevant from a users' point of view are depicted and selected based on a usage context analysis. This is facilitated by providing factors of the mobile usage context and a catalog of 50 interaction scenarios that are potentially architectural sensitive (derived from patterns, referring to them, and complying to a scenario life cycle). The analysis of the architecture itself is based on the principle of view models. We analyze, in what way structure or behavior of architectural elements can constrain the interaction described by a specific interaction scenario. Results verify that usability was considered architecturally, explain disadvantageous and advantageous architectural decisions as well as trade-offs with other quality factors. Thus, results can flow back to architectural design and usability engineering. Additionally, the method is combined with a user test.

With this research, analyzing and assessing the architectural support for the usability of mobile applications is easier than with earlier works. This inspires further research, for example, more case studies regarding the relationship between usability and software architecture, other quality factors and software architecture, new possibilities for construction in agile processes, and the cooperation amongst the fields usability engineering and software engineering in general.

DISSERTATIONSBEZOGENE BIBLIOGRAPHISCHE DATEN

Die folgenden Artikel wurden im Rahmen der Dissertation veröffentlicht und in dieser Arbeit zitiert.

- Bettina Biel, Stefan Seitz, and Volker Gruhn, *Towards pattern-based generic interaction scenarios*, Workshop Frontiers in Ambient and Mobile Systems, Proc. of the 8th Mobile Web Information Systems MobiWIS, 2011. [BSG11]
- Bettina Biel and Volker Gruhn, *Usability-improving mobile application development patterns*, Proc. of the 15th European Conference on Pattern Languages of Programs (EuroPLoP 2010), July 7-11, 2010, Irsee Monastery, Bavaria, Germany, 2010. [BG10b]
- Bettina Biel and Volker Gruhn, *Analyzing the architectural support of usability*, 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010), 2010, pp. 20–27. [BG10a]
- Bettina Biel, Thomas Grill, and Volker Gruhn, *Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application*, Journal of Systems and Software **83** (2010), no. 11, pp. 2031 – 2044, Interplay between Usability Evaluation and Software Development. [BGG10]
- Bettina Biel and Volker Gruhn, *Towards a method for analyzing architectural support levels of usability*, Proc. of the European Conference on Software Architecture (ECSA/WICSA 2009), 2009, pp. 273–276. [BG09]
- Thomas Grill, Bettina Biel, and Volker Gruhn, *A pattern approach to mobile interaction design*, it - Information Technology **51** (2009), no. 2, pp. 93–101. [GBG09]
- Bettina Biel, Thomas Grill, and Volker Gruhn, *Patterns of trust*, Workshop Trustworthy Ubiquitous Computing, Proc. of the Mobile Multimedia (MoMM 2008), November 2008, pp. 391–396. [BGG08] (Best Workshops Paper Award)
- Bettina Biel and Volker Gruhn, *Content adaptation*, Proc. of the VikingPLoP, 2007. [BG07]

INHALTSVERZEICHNIS

1 EINLEITUNG 1
- 1.1 Motivation . . . 1
- 1.2 Architektonische Unterstützung von Usability . . . 2
- 1.3 Frühere Arbeiten . . . 3
- 1.4 Probleme der früheren Arbeiten . . . 4
- 1.5 Vorgehen in diesem Forschungsprojekt . . . 4
- 1.6 Die Methode SATURN . . . 5
- 1.7 Verbesserungen durch die Methode SATURN . . . 6
- 1.8 Überblick über die Kapitel . . . 6

2 FRÜHERE ARBEITEN 8
- 2.1 Softwarearchitekturanalysemethoden . . . 8
- 2.2 Szenario-basierte Softwarearchitekturanalysemethoden . . . 9
- 2.3 Qualitätsanforderungen an szenario-basierte Softwarearchitekturanalysemethoden . . . 10
- 2.4 Beschreibung der Methode SALUTA . . . 11
  - 2.4.1 Kurzvorstellung der Methode SALUTA . . . 11
  - 2.4.2 Detailliertes Vorgehen in der Methode SALUTA . . . 12
- 2.5 Beschreibung der Methode ATAM . . . 15
  - 2.5.1 Kurzvorstellung der Methode ATAM . . . 15
  - 2.5.2 Detailliertes Vorgehen in der Methode ATAM . . . 16
- 2.6 Bewertung der Methoden SALUTA und ATAM . . . 20
- 2.7 Forschungsfragen . . . 25
- 2.8 Zusammenfassung . . . 25

3 INTERAKTIONSSZENARIOS FÜR DIE METHODE SATURN 27
- 3.1 Motivation . . . 27
- 3.2 Verwandte Arbeiten . . . 27
  - 3.2.1 Szenarios . . . 27
  - 3.2.2 Erhebung von Szenarios . . . 30
- 3.3 Definition und Klassifikation der Interaktionsszenarios . . . 31
  - 3.3.1 Definition eines Interaktionsszenarios . . . 31
  - 3.3.2 Klassifikation nach Interaktionskategorien . . . 33
  - 3.3.3 Klassifikation nach Usability-Ziel . . . 34
  - 3.3.4 Klassifikation nach möglichen Architekturänderungen . . . 36
- 3.4 Erhebung der Interaktionsszenarios . . . 39
  - 3.4.1 Extraktionsmethode und Beispiel . . . 39
  - 3.4.2 Patterns . . . 40
  - 3.4.3 Ausgewählte Pattern-Sammlungen . . . 41
  - 3.4.4 Beispiel . . . 42
- 3.5 Lebenszyklus der pattern-basierten Interaktionsszenarios . . . 44
- 3.6 Zusammenfassung . . . 46

4 DIE ERSTELLUNG DER METHODE SATURN 48
- 4.1 Erstellen der Methode SATURN . . . 48
  - 4.1.1 Struktur und Bestandteile einer Methode . . . 48
  - 4.1.2 Forschungsfrage 1: Wie und wann mobilen Nutzungskontext integrieren? . . . 48
  - 4.1.3 Forschungsfragen 2 und 3: Welches Format für Szenarios und wie Usability operationalisieren? . . . 54
  - 4.1.4 Forschungsfrage 4: Wie Szenarios fachlich motiviert auswählen? . . . 55
  - 4.1.5 Forschungsfrage 5: Hilfsmittel ohne Pattern-Expertise? . . . 55

4.1.6 Forschungsfrage 6: Analyse offenlegende Fragetechnik mit geringeren Freiheitsgraden als frühere Arbeiten? . . . 55
4.1.7 Forschungsfrage 7, Teil 1: Wie voraussichtlichen Einfluss von Architekturentscheidungen auf Usability bestimmen und bewerten? 56
4.1.8 Forschungsfrage 7, Teil 2: Wie potenziellen Änderungsaufwand der Softwarearchitektur bestimmen und bewerten? . . . 57
4.1.9 Forschungsfrage 8: Wie iterative Erstellung der Architektur unterstützen? . . . 58
4.1.10 Forschungsfrage 9: Wie interdisziplinäre Kooperation unterstützen? 61
4.2 Beschreibung der Methode SATURN . . . 62
4.2.1 Metamodell . . . 62
4.2.1.1 Qualitätsmodell von SATURN . . . 62
4.2.1.2 Spezifikation von Usability-Anforderungen in SATURN . 63
4.2.1.3 Argumentationspfad . . . 64
4.2.1.4 Beschreibung der Architektur . . . 64
4.2.2 Kurzübersicht von SATURN . . . 67
4.2.3 SATURN-Phase 1: Beschreiben des Analysekontexts . . . 68
4.2.4 SATURN-Phase 2: Bestimmen der Interaktionsszenarios . . . 68
4.2.5 SATURN-Phase 3: Evaluieren der Interaktionsszenarios . . . 69
4.2.6 SATURN-Phase 4: Interpretieren der Resultate . . . 71
4.2.7 SATURN-Phase 5: Retrospektive . . . 73
4.3 Kombination von SATURN mit einem Nutzertest . . . 74
4.4 Antworten auf Forschungsfragen . . . 76
4.5 Zusammenfassung . . . 82
5 FALLSTUDIEN 84
5.1 Studiendesign . . . 84
5.1.1 Canonical Action Research . . . 84
5.1.2 Canonical Action Research und SATURN . . . 85
5.1.3 Kurzvorstellung der Fallstudien . . . 86
5.2 Fallstudie „Shake Your Mac“ . . . 86
5.2.1 Diagnose . . . 86
5.2.2 Planung . . . 86
5.2.3 Durchführung der Fallstudie SYM: SATURN-Phase 1 . . . 87
5.2.3.1 Name und Verwendungszweck . . . 87
5.2.3.2 Nutzungskontext . . . 87
5.2.3.3 Beschreiben der Softwarearchitektur . . . 88
5.2.4 Durchführung der Fallstudie SYM: SATURN-Phase 2 . . . 91
5.2.4.1 Mapping der Interaktionsszenarios zu Anforderungen . 91
5.2.4.2 Auswahl aus CASSINI . . . 91
5.2.4.3 Bestimmen der zu evaluierenden Interaktionsszenarios . 91
5.2.5 Durchführung der Fallstudie SYM: SATURN-Phase 3 . . . 93
5.2.5.1 Canceling Commands (Abbrechen, Cancel) . . . 93
5.2.5.2 Add New Input/Output Device . . . 95
5.2.5.3 Observing System State (System Feedback) . . . 97
5.2.5.4 Recovering from Failure . . . 99
5.2.6 Durchführung der Fallstudie SYM: SATURN-Phase 4 . . . 101
5.2.6.1 Architektur-Support-Level . . . 101
5.2.6.2 Architekturentscheidungen auflisten . . . 101
5.2.6.3 Themen beschreiben . . . 104
5.2.6.4 Sonstiges . . . 108
5.2.7 Durchführung der Fallstudie SYM: SATURN-Phase 5 . . . 109
5.2.8 Nutzertest . . . 111
5.2.8.1 Durchführung des Nutzertests . . . 111

5.2.8.2 Resultate des Nutzertests . . . 112
5.2.8.3 Resultate von SATURN und Nutzertest . . . 114
5.2.8.4 Rückblick . . . 118
5.2.9 Beurteilung . . . 120
5.2.9.1 Machbarkeit . . . 120
5.2.9.2 Nützlichkeit . . . 120
5.2.9.3 Verständnis . . . 121
5.2.10 Reflexion . . . 121
5.2.10.1 Aufwand . . . 121
5.2.10.2 Aufbau und Inhalte der Methode . . . 122
5.2.10.3 Wechselwirkungen . . . 122
5.3 Fallstudie „Traffic Scanner“ . . . 123
5.3.1 Diagnose . . . 123
5.3.2 Planung . . . 123
5.3.3 Durchführung der Fallstudie TS: SATURN-Phase 1 . . . 123
5.3.3.1 Name und Hauptverwendungszweck . . . 123
5.3.3.2 Nutzungskontext . . . 124
5.3.3.3 Mapping von Interaktionsklassen auf Eigenschaften des Nutzungskontexts . . . 126
5.3.3.4 Beschreiben der Softwarearchitektur . . . 126
5.3.4 Durchführung der Fallstudie TS: SATURN-Phase 2 . . . 130
5.3.4.1 Vorauswahl von Interaktionsszenarios aus CASSINI . . . 130
5.3.4.2 Diskussion und Bestimmen der zu evaluierenden Interaktionsszenarios . . . 130
5.3.5 Durchführung der Fallstudie TS: SATURN-Phase 3 . . . 133
5.3.5.1 System-Feedback . . . 134
5.3.5.2 Abbrechen . . . 135
5.3.5.3 Forgiving Format . . . 136
5.3.5.4 Checking for Correctness . . . 137
5.3.5.5 Context-based Workflow . . . 139
5.3.5.6 Using Applications Concurrently . . . 141
5.3.5.7 Maintaining Device Independence . . . 142
5.3.5.8 Progress Indication . . . 143
5.3.5.9 Familiar Appearance and/or Behavior . . . 144
5.3.5.10 Verifying Resources . . . 145
5.3.5.11 Error Messages . . . 146
5.3.5.12 Help . . . 148
5.3.6 Durchführung der Fallstudie TS: SATURN-Phase 4 . . . 149
5.3.6.1 Architektur-Support-Level . . . 149
5.3.6.2 Architekturentscheidungen auflisten . . . 150
5.3.6.3 Themen beschreiben . . . 152
5.3.6.4 Priorisierung der Interaktionsszenarios basierend auf Nutzungskontext . . . 161
5.3.6.5 Resultierende ASL-Tabelle . . . 161
5.3.7 Durchführung der Fallstudie TS: SATURN-Phase 5 . . . 162
5.3.7.1 Feedback . . . 162
5.3.7.2 Pflege der Wissensbasis . . . 163
5.3.8 Beurteilung . . . 163
5.3.8.1 Machbarkeit . . . 163
5.3.8.2 Nützlichkeit . . . 164
5.3.8.3 Verständnis . . . 164
5.3.9 Reflexion . . . 164
5.3.9.1 Aufwand . . . 164

5.3.9.2 Aufbau und Inhalte der Methode . . . 165
5.3.9.3 Wechselwirkungen . . . 165
5.4 Zusammenfassung . . . 165
6 VALIDATION DER METHODE SATURN 167
6.1 Validationsziele . . . 167
6.2 Konstruktive Validität . . . 169
6.2.1 Forschungsmethodik . . . 169
6.2.2 Definitionen der einzelnen theoretischen Elemente . . . 169
6.2.3 Zusammenhänge zwischen den theoretischen Elementen . . . 169
6.2.4 Vorgaben für die Bewertungen . . . 170
6.2.5 Wahl der Skalen . . . 170
6.2.6 Zusammenfassung zur konstruktiven Validität . . . 171
6.3 Interne Validität . . . 171
6.3.1 Theoretische Nachvollziehbarkeit der Resultate . . . 171
6.3.2 Ergebnisse der Fallstudien . . . 173
6.3.3 Zusammenfassung der Ergebnisse . . . 185
6.3.4 Das Ergebnis verwischende Variablen . . . 190
6.3.5 Zusammenfassung zur internen Validität . . . 190
6.4 Externe Validität . . . 190
6.5 Nützlichkeit . . . 191
6.5.1 Ziele von Softwarearchitekturanalysemethoden . . . 191
6.5.2 Zielerreichungsgrad in Fallstudien . . . 192
6.5.3 Aufwand und Nutzen der betrachteten Methoden . . . 192
6.5.3.1 Ressourcen- und Zeitaufwand . . . 192
6.5.3.2 Nutzen . . . 193
6.5.3.3 Fazit . . . 193
6.5.4 Allgemeine Kritik an szenario-basierten Methoden . . . 194
6.5.4.1 Szenario-Abdeckung . . . 195
6.5.4.2 Ressourcen- und Zeitaufwand . . . 195
6.5.4.3 Schulungsunterlagen . . . 195
6.5.4.4 Validation mit Fallstudien . . . 196
6.5.4.5 Werkzeugunterstützung . . . 196
6.5.4.6 Abhängigkeit von Experten . . . 196
6.5.4.7 Verständnis der Spezifikationen bei Softwarearchitekten 197
6.5.4.8 Softwarequalität . . . 197
6.5.4.9 Fazit . . . 197
6.6 Zusammenfassung . . . 197
7 FAZIT 199
7.1 Wesentliche Beiträge der Arbeit . . . 199
7.1.1 Architekturanalyse im Anforderungskontext . . . 199
7.1.2 Koordination mit dem Usability Engineering . . . 199
7.1.3 Interaktionsszenarios . . . 200
7.2 Weiterführende Forschung . . . 201

I APPENDIX 202
A ANHANG 203
A.1 Vorlage für ein Interaktionsszenario . . . 203
A.2 Katalog der Interaktionsszenarios . . . 204
A.3 Qualitätsmerkmale . . . 255

LITERATURVERZEICHNIS 256

ABBILDUNGSVERZEICHNIS

Abbildung 1 Nutzungsprofil der Fallstudie der Anwendung eSuite [FB05] . . 12
Abbildung 2 Software-Architecture-Usability-Framework [FGB03] 13
Abbildung 3 Ergebnisse der Fallstudie Webplatform aus [FvGB04] 14
Abbildung 4 Ergebnisdarstellung in SALUTA, Tabelle aus [FB05] 14
Abbildung 5 Usability-Taktiken [BCK03] 18
Abbildung 6 Liste mit Verantwortlichkeiten für das Szenario Cancel [GJB05] . 20
Abbildung 7 Interaktionsszenario 31
Abbildung 8 Zusammenhänge zwischen essentiellen Pattern-Abschnitten . . . 40
Abbildung 9 Screenshot des Katalogs der Interaktionsszenarios (letzter Aufruf 16.05.2011) 44
Abbildung 10 Datenbankmodell Interaktionsszenariokatalog 44
Abbildung 11 Aktivitätsdiagramm Interaktionszenario-Lebenszyklus 45
Abbildung 12 Aktivitätsdiagramm Pattern-Lebenszyklus 46
Abbildung 13 Qualitätsmodell von SATURN 62
Abbildung 14 Usability spezifizieren in SATURN 63
Abbildung 15 Argumentationspfad in SATURN 64
Abbildung 16 Aktivitätsdiagramm zu SATURN 67
Abbildung 17 Zyklus der Anwendungsnahen Forschung nach [DMK04] 85
Abbildung 18 Nutzungskontext SYM 88
Abbildung 19 Komponentendiagramm von SYM 88
Abbildung 20 CRC-Karten von SYM 89
Abbildung 21 Sequenzdiagramm von SYM 90
Abbildung 22 SYM – Auswahl der Interaktionsszenarios 92
Abbildung 23 Evaluationsformular Cancel, Seite 1 93
Abbildung 24 Evaluationsformular Cancel, Seite 2 94
Abbildung 25 Evaluationsformular Add New Input/Output Device, Seite 1 . . 95
Abbildung 26 Evaluationsformular Add New Input/Output Device, Seite 2 . . 96
Abbildung 27 Evaluationsformular Feedback, Seite 1 97
Abbildung 28 Evaluationsformular Feedback, Seite 2 98
Abbildung 29 Evaluationsformular Recovering from Failure, Seite 1 99
Abbildung 30 Evaluationsformular Recovering from Failure, Seite 2 100
Abbildung 31 Tabelle Architektur-Support-Level „Shake Your Mac“ 101
Abbildung 32 SYM-Architekturentscheidungen, Seite 1 102
Abbildung 33 SYM-Architekturentscheidungen, Seite 2 103
Abbildung 34 S1, S2, S8, S9 betreffen Kommunikation 104
Abbildung 35 Entscheidungen S3, S6, S7 betreffen das Thema Antwortverhalten 106
Abbildung 36 Entscheidungen S4 und S5 betreffen das Thema Erweiterbarkeit . 107
Abbildung 37 Während der Softwarearchitekturanalyse wurden weitere Kommentare notiert. 109
Abbildung 38 Bewertung der Interaktionsszenarios und der darin gestellten Fragen an eine Usability-Evaluation 109
Abbildung 39 Analyse der Evaluation mit MacEval 111
Abbildung 40 Verteilung der Interaktionstechniken bei der Interaktion mit PowerPoint und Google Earth 113
Abbildung 41 Mapping der Ergebnisse von User-Evaluation und SATURN, Seite 1 117
Abbildung 42 Mapping der Ergebnisse von User-Evaluation und SATURN, Seite 2 118

Abbildung 43 Ergebnisse für zukünftige Entwicklungszyklen 119
Abbildung 44 Rahmenbedingungen von TrafficScanner 127
Abbildung 45 TrafficScanner-Softwarearchitektur im Überblick 127
Abbildung 46 Schnittstellenbeschreibung mit Schnittstellen zwischen Klassen/Komponenten sowie Format und Medium 129
Abbildung 47 Ergebnis der Szenario-Auswahl, Seite 1 131
Abbildung 48 Ergebnis der Szenario-Auswahl, Seite 2 132
Abbildung 49 ASL-Tabelle für die Analyse von TrafficScanner 133
Abbildung 50 Analyseformular System-Feedback 134
Abbildung 51 Analyseformular Abbrechen 135
Abbildung 52 Analyseformular Forgiving Format 136
Abbildung 53 Analyseformular Checking for Correctness, Seite 1 137
Abbildung 54 Analyseformular Checking for Correctness, Seite 2 138
Abbildung 55 Analyseformular Context-based Workflow, Seite 1 139
Abbildung 56 Analyseformular Context-based Workflow, Seite 2 140
Abbildung 57 Analyseformular Using Applications Concurrently 141
Abbildung 58 Analyseformular Maintaining Device Independence 142
Abbildung 59 Analyseformular Progress Indication 143
Abbildung 60 Analyseformular Familiar Appearance and/or Behavior 144
Abbildung 61 Analyseformular Verifying Resources 145
Abbildung 62 Analyseformular Error Messages, Seite 1 146
Abbildung 63 Analyseformular Error Messages, Seite 2 147
Abbildung 64 Analyseformular Help 148
Abbildung 65 ASL TrafficScanner 149
Abbildung 66 szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 1/3 150
Abbildung 67 szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 2/3 151
Abbildung 68 szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 3/3 152
Abbildung 69 Entscheidungen S1, S5, S6, S9 und S14 betreffen die Staumeldungen 153
Abbildung 70 S2 betrifft das Thema Abbrechen 155
Abbildung 71 S3 und S4 betreffen das Thema Login-Probleme 156
Abbildung 72 S7 betrifft das Thema Gleichzeitige Nutzung von Programmen . 157
Abbildung 73 S8 betrifft das Thema Externe GPS-Geräte ergänzen 158
Abbildung 74 S10 betrifft das Thema Plattformspezifische Styleguides 158
Abbildung 75 S11 betrifft das Thema Gesicherter Start der Anwendung 159
Abbildung 76 S12, S13, S14, S15 betreffen das Thema Fehlermeldungen 159
Abbildung 77 S16 betrifft das Thema Kontext-sensitive Hilfe 160
Abbildung 78 Fragen an Usability-Evaluation TrafficScanner 161
Abbildung 79 Finale ASL-Tabelle für TrafficScanner 162
Abbildung 80 von Interaktionsszenarios über Use Cases zu Architekturentscheidungen (SYM) 188
Abbildung 81 von Interaktionsszenarios über Use Cases zu Architekturentscheidungen (TS) 189

Tabelle 1 Generierungstabelle für Usability-Attribut-Szenarios [BCK03] . . 17
Tabelle 2 Inhaltliche Betrachtung der Methoden ATAM und SALUTA . . . 24
Tabelle 3 Essential Use Cases und dazu gehörender Detailed Use Case ([CL99] zitiert in [FJ06]) 29
Tabelle 4 Kernelemente eines Interaktionsszenarios 32
Tabelle 5 Elemente, die ein Interaktionsszenario klassifizieren 33
Tabelle 6 Verteilung der Interaktionsszenarios über Interaktionskategorien 34
Tabelle 7 Usability-Definitionen im Vergleich 35
Tabelle 8 Verteilung der Usability-Attribute in Interaktionsszenarios 36
Tabelle 9 Komplexität von architektonischen Änderungen 37
Tabelle 10 Vertrauensfaktor für Bewertungen 37
Tabelle 11 Potenzielle Architektursensitivität der Interaktionsszenarios . . . 38
Tabelle 12 Extraktion der Beschreibung des Interaktionsszenarios Cancel (Abbrechen) aus dem Pattern Cancel [Tid05] 42
Tabelle 13 Interaktionsszenario Abbrechen (Cancel) 43
Tabelle 14 Kontextfaktor Umgebung 50
Tabelle 15 Kontextfaktor Benutzer 51
Tabelle 16 Kontextfaktor Aufgabe 52
Tabelle 17 Kontextfaktor Endgerät 53
Tabelle 18 Kontextfaktor Mobile Anwendung 54
Tabelle 19 Qualitätsmodelle im Überblick (inhaltlich ähnliche Faktoren in einer Zeile) 61
Tabelle 20 Einfluss einer Architekturentscheidung auf Usability 70
Tabelle 21 Komplexität von architektonischen Änderungen 70
Tabelle 22 Architektur-Support-Level-Tabelle (ASL-Tabelle) 71
Tabelle 23 Schema für Themen 72
Tabelle 24 Vergleich der Ergebnisse von SA-Analyse (SAA) und User-Evaluation (UE) 74
Tabelle 25 Einfluss einer Architekturentscheidung auf Usability 78
Tabelle 26 Komplexität von architektonischen Änderungen 78
Tabelle 27 Architektur-Support-Level-Tabelle mit Bewertungen und Prioritäten der Interaktionsszenarios 79
Tabelle 28 Bewertung der Methoden ATAM, SALUTA und SATURN (Teil 1/2) 81
Tabelle 29 Bewertung der Methoden ATAM, SALUTA und SATURN (Teil 2/2) 82
Tabelle 30 Beobachtete Parameter 112
Tabelle 31 Varianzanalyse der Nutzungskontextfaktoren 114
Tabelle 32 Vergleich der Resultate einer Software-Architektur-Analyse und der Usability-Evaluation 115
Tabelle 33 Hauptzielgruppen des TrafficScanners: Berufsautofahrer und Rentner, die viel und/oder gerne fahren 124
Tabelle 34 Beispiel-Mapping von Nutzungskontext und Interaktionskategorien der Fallstudie TrafficScanner 126
Tabelle 35 Clustering von Architekturentscheidungen in Themen 152
Tabelle 36 Übersicht über die Verwendung von Interaktionsszenarios 163
Tabelle 37 Aspekte von Validität und Zielvorgabe dieses Forschungsprojektes [BM85] 168

Tabelle 38 Schema für Argumentationspfade . . . 172
Tabelle 39 Argumentationspfad des Interaktionsszenarios Abbrechen in Fallstudie SYM . . . 173
Tabelle 40 Argumentationspfad des Interaktionsszenarios Multi-Channel Access in Fallstudie SYM . . . 174
Tabelle 41 Argumentationspfad des Interaktionsszenarios System-Feedback in Fallstudie SYM . . . 175
Tabelle 42 Argumentationspfad des Interaktionsszenarios Wiederherstellung nach Betriebsausfall in Fallstudie SYM . . . 176
Tabelle 43 Argumentationspfad des Interaktionsszenarios System-Feedback in Fallstudie TS . . . 177
Tabelle 44 Argumentationspfad des Interaktionsszenarios Abbrechen in Fallstudie TS . . . 177
Tabelle 45 Argumentationspfad des Interaktionsszenarios Nachsichtiges Format in Fallstudie TS . . . 178
Tabelle 46 Argumentationspfad des Interaktionsszenarios Prüfen auf Korrektheit in Fallstudie TS . . . 178
Tabelle 47 Argumentationspfad des Interaktionsszenarios Kontextbewusste Interaktion in Fallstudie TS . . . 179
Tabelle 48 Argumentationspfad des Interaktionsszenarios Konfliktfreie Nebenläufigkeit in Fallstudie TS . . . 180
Tabelle 49 Argumentationspfad des Interaktionsszenarios Geräteunabhängigkeit in Fallstudie TS . . . 180
Tabelle 50 Argumentationspfad des Interaktionsszenarios Fortschrittsanzeige in Fallstudie TS . . . 181
Tabelle 51 Argumentationspfad des Interaktionsszenarios Vertrautes Aussehen und Verhalten in Fallstudie TS . . . 181
Tabelle 52 Argumentationspfad des Interaktionsszenarios Prüfen notwendiger Ressourcen in Fallstudie TS . . . 182
Tabelle 53 Argumentationspfad des Interaktionsszenarios Fehlermeldungen in Fallstudie TS . . . 183
Tabelle 54 Argumentationspfad des Interaktionsszenarios Mehrstufige Hilfe in Fallstudie TS . . . 184
Tabelle 55 Bewertung der Interaktionsszenarios in Fallstudien SYM, (Behinderung der Response, Einfluss auf Architektur) . . . 185
Tabelle 56 Szenario-beeinflussende Entscheidungen (S) der Fallstudie Shake Your Mac (SYM) und ihre Klassifikation . . . 185
Tabelle 57 Bewertung der Interaktionsszenarios in Fallstudie TS (Behinderung der Response, Einfluss auf Architektur) . . . 186
Tabelle 58 Szenario-beeinflussende Entscheidungen (S) der Fallstudie TrafficScanner (TS) und ihre Klassifikation . . . 187
Tabelle 59 Szenario-beeinflussende Entscheidungen (S) der Fallstudien SYM und TS und ihre Klassifikation . . . 187
Tabelle 60 Aufwand und Nutzen . . . 194
Tabelle 61 Generierungstabelle für ein Interaktionsszenario . . . 203
Tabelle 62 Interaktionsszenario System-Feedback . . . 205
Tabelle 63 Interaktionsszenario Aktionen auf mehreren Objekten (Actions on Multiple Objects) . . . 206
Tabelle 64 Interaktionsszenario Makros (Macros) . . . 207
Tabelle 65 Interaktionsszenario Abbrechen (Cancel) . . . 208
Tabelle 66 Interaktionsszenario Nachsichtiges Formular (Forgiving Format) 209
Tabelle 67 Interaktionsszenario Strukturiertes Format (Structured Format) . 210
Tabelle 68 Interaktionsszenario Eingabefenster (Input Widgets) . . . 211

Tabelle 69 Interaktionsszenario Gleiche oder Separate Ansicht (Same or Separate Screen, Pagination) . . . 212
Tabelle 70 Interaktionsszenario Lückenfüller (Fill-in-the-Blanks) . . . 213
Tabelle 71 Interaktionsszenario Vor-Ort-Hinweis (Hint-in-Place) . . . 214
Tabelle 72 Interaktionsszenario Gute Voreinstellungen (Good Defaults) . . . 215
Tabelle 73 Interaktionsszenario Auto-Komplettierung (Auto-Completion) . 216
Tabelle 74 Interaktionsszenario Sofortige Gültigkeitsprüfung (Instant Validation) . . . 217
Tabelle 75 Interaktionsszenario Prüfen auf Korrektheit (Checking for Correctness) . . . 218
Tabelle 76 Interaktionsszenario Aufforderung zur Fehlerbehebung (Prompt for Error Correction) . . . 219
Tabelle 77 Interaktionsszenario Leistungsabhängige Listenlänge (Result List Length) . . . 220
Tabelle 78 Interaktionsszenario Befehlsprotokoll (Command History) . . . 221
Tabelle 79 Interaktionsszenario Alternativansichten (Alternative Views) . . . 222
Tabelle 80 Interaktionsszenario Modi und Profile (Modes and Profiles) . . . 223
Tabelle 81 Interaktionsszenario Internationalisierung (Supporting International Use) . . . 224
Tabelle 82 Interaktionsszenario Mehrkanalzugang (Multi-Channel Access) . 225
Tabelle 83 Interaktionsszenario Mehrfach-Widerruf (Multi-Level Undo) . . . 226
Tabelle 84 Interaktionsszenario Workflow-Modell (Workflow Model) . . . 227
Tabelle 85 Interaktionsszenario Konfliktfreie Nebenläufigkeit (Non-conflicting Application Concurrency) . . . 228
Tabelle 86 Interaktionsszenario Geräteunabhängigkeit (Device Independence, ehem. Non-conflicting Device Usage) . . . 229
Tabelle 87 Interaktionsszenario Wiederherstellung nach Betriebsausfall (Recovering From Failure) . . . 230
Tabelle 88 Interaktionsszenario Vergessene Passwörter zurückbekommen (Retrieving Forgotten Passwords . . . 231
Tabelle 89 Interaktionsszenario Weiterverwenden von Informationen (Reusing Information) . . . 232
Tabelle 90 Interaktionsszenario Varianten des Einfügens (Paste Variations) . 233
Tabelle 91 Interaktionsszenario Multitasking (Supporting Multiple Activities) . . . 234
Tabelle 92 Interaktionsszenario Benutzer-Tempo (User's Pace) . . . 235
Tabelle 93 Interaktionsszenario Fortschrittsanzeige (Progress Indication) . . 236
Tabelle 94 Interaktionsszenario Umfassende Suche (Comprehensive Searching) . . . 237
Tabelle 95 Interaktionsszenario Vertrautes Aussehen und Verhalten (Familiar Appearance and/or Behaviour) . . . 238
Tabelle 96 Interaktionsszenario Wechsel zwischen neuem und altem User Interface (Switching between New and Traditional Interfaces) . . 239
Tabelle 97 Interaktionsszenario Wortreiches User Interface (Verbose Interface) 240
Tabelle 98 Interaktionsszenario Prüfen notwendiger Ressourcen (Verifying Resources) . . . 241
Tabelle 99 Interaktionsszenario Gleiche Bedienung verschiedener Ansichten (Operating Consistently Across Views) . . . 242
Tabelle 100 Interaktionsszenario Ansichten verfügbar machen (Making views accessible, war: Consistent Set of Views) . . . 243
Tabelle 101 Interaktionsszenario Strukturieren und Verstecken von Content (Structuring and Hiding Content) . . . 244

Tabelle 102 Interaktionsszenario Bewegbare Panels (Movable Panels, war: Amendable Screen Layout) . . . 245
Tabelle 103 Interaktionsszenario Dynamische Abfrage (Dynamic Query) . . . 246
Tabelle 104 Interaktionsszenario Fehlermeldungen (Error Message) . . . 247
Tabelle 105 Interaktionsszenario Wizard . . . 248
Tabelle 106 Interaktionsszenario Allmähliche Offenlegung (Responsive Disclosure/Enabling) . . . 249
Tabelle 107 Interaktionsszenario Extras auf Wunsch (Extras on Demand) . . . 250
Tabelle 108 Interaktionsszenario Mehrstufige Hilfe (Multi-Level Help) . . . 251
Tabelle 109 Interaktionsszenario Sortierbare Tabelle (Sortable Table, war: Retrieving Information in a Table) . . . 252
Tabelle 110 Interaktionsszenario Kontextbewusste Interaktion (Context-aware Interaction) . . . 253
Tabelle 111 Interaktionsszenario Neustart einer Anwendung (Re-entering an Application) . . . 254
Tabelle 112 Qualitätsmodelle im Überblick (inhaltlich ähnliche Faktoren in einer Zeile) . . . 255

ACRONYME

AE	Architekturentscheidungen
AR	Action Research, Anwendungsnahe Forschung
ASL	Architektur-Support-Level
ATAM	„Architecture Tradeoff Analysis Method“
CAR	Canonical Action Research
CASSINI	„CAtalog of potentially architecture-SenSitive INteraction scenarIos“
ESA	Erstellung einer Softwarearchitektur
IA	Informationsarchitektur
MMI	Mensch-Maschine-Interaktion
S	szenario-beeinflussende Architekturentscheidung
SA	Softwarearchitektur
SAA	Softwarearchitekturanalysemethode
SALUTA	„Scenario-based Architecture-Level UsabiliTy Analysis“
SATURN	„SoftwareArchiTekturanalyse von Usability-anfoRderungeN“
SQM	Softwarequalitätsmerkmale
SYM	Shake Your Mac
TS	TrafficScanner
UI	User Interface

UML Unified Modeling Language

USE Usability Engineering

1 EINLEITUNG

1.1 MOTIVATION

Rasante Innovationszyklen, kurze Produkteinführungszeiten und ein hoher Konkurrenzdruck sind typische Rahmenbedingungen für die Entwicklung mobiler Anwendungen. Dies sind Anwendungen, die auf mobilen Endgeräten laufen und in verschiedenen Umgebungen verwendet werden. Es ist eine Herausforderung, immer wieder neue Interaktionstechnologien und Funktionalitäten verschiedener Endgeräte zu integrieren und dabei die hohen Erwartungen der Nutzer zu erfüllen: eine mobile Anwendung muss einfach benutzbar sein.

Der Begriff Usability (Benutzbarkeit) steht für die Qualität bei der Benutzung eines Produktes [Bev01] und beschreibt, inwiefern ein Produkt durch bestimmte Personen verwendet werden kann, damit sie bestimmte Ziele in einem bestimmten Nutzungskontext erreichen können. Usability-Attribute sind Nützlichkeit, Erlernbarkeit, Einprägsamkeit, Effizienz, Effektivität und Sicherheit [PRS07]. Usability ist wie Ästhetik, Spaß und Zufriedenheit ein Teilziel der User Experience, des gesamten Benutzungserlebnisses [PRS07].

Häufig wird Usability bei der Softwareentwicklung erst dann thematisiert, wenn ein lauffähiger Prototyp oder eine fertige Software vorliegen. Das kann dazu führen, dass Probleme bei der Benutzung (Usability-Probleme) auftreten. Usability-Probleme verwirren und behindern Nutzer und führen zu niedrigen Umsätzen, aufwendigen Schulungen, teurem Kundenservice, Image-Verlust [Nie93] und langfristig hohen Wartungskosten [FB03]. Um Usability-Probleme frühzeitig zu erkennen, involvieren benutzerzentrierte Design-Prozesse die Benutzer so häufig wie möglich und integrieren interdisziplinäres Wissen [PRS07] aus Bereichen wie Psychologie, Ergonomie, Grafikdesign und Softwaretechnik. Das Usability Engineering ist solch ein benutzerzentrierter Design-Prozess von Software, in dem Anforderungserhebung, Design und Evaluation zyklisch und iterativ aufeinander folgen [Nie93]. Besonders schwerwiegend sind Usability-Probleme, die nur behoben werden können, wenn die Softwarearchitektur geändert wird [BCK03, FvWB06].

Es gibt viele Möglichkeiten, den Begriff Softwarearchitektur zu definieren [SEI12]. Nach Kruchten [Kru98] ist eine Softwarearchitektur (SA, Architektur) die Menge der Entscheidungen darüber, wie ein Softwaresystem organisiert ist, die Auswahl der strukturellen Elemente und ihrer Schnittstellen, durch die das System komponiert wird, ihr Verhalten, welches durch die Kollaborationen zwischen den Elementen spezifiziert wird, die Komposition dieser Elemente in progressiv größere Subsysteme und der architektonische Stil, der diese Konstruktion leitet. [Kru98]

Die Softwarearchitektur setzt die Rahmenbedingungen für die Qualitätsmerkmale wie Performanz, Modifizierbarkeit und Sicherheit. Softwarearchitekturentscheidungen sind besonders kritisch, weil Modifikationen an einem architektonischen Element oft Änderungen, Fehlerquellen und Probleme an anderen Elementen nach sich ziehen [BCK03]. Architekturentscheidungen, die zu Architekturänderungen führen, werden als „architektursensitiv“ bezeichnet. [BCK03, FvWB06]

Bei der Erstellung der Softwarearchitektur, einem iterativen Prozess, werden Anforderungen auf Softwarekomponenten abgebildet [Som01, Nus01, PBG04]. Eine Anforderung ist dabei eine vereinbarte und dokumentierte Bedingung oder Eigenschaft, die ein System oder eine Person benötigt, um ein Problem zu lösen oder ein Ziel zu erreichen [IEE90, Poh06]. Anforderungserhebung, Design und Analyse werden bei der

Erstellung der Softwarearchitektur zyklisch durchlaufen. Anforderungen an Funktionalität, Qualität und Rahmenbedingungen der Software schränken den Lösungsraum mehr und mehr ein, bis ein oder mehrere alternative Architekturen entstehen. Die Erstellung der Softwarearchitektur ist eine anspruchsvolle Aufgabe, bei der durch neue Ideen, Probleme und Austauschbeziehungen zwischen einzelnen Architekturentscheidungen neue Anforderungen definiert oder bestehende angepasst werden müssen [Nus01].

Mit einer Softwarearchitekturanalysemethode (SAA, Architekturanalyse) wird die architektonische Unterstützung eines oder mehrerer Qualitätsmerkmale analysiert. Softwarearchitekturanalysemethoden beschreiben Aspekte einer Softwarearchitektur und ermitteln Probleme, um neue oder geänderte Anforderungen zu identifizieren [Nus01, BCK03, Poh06]. Architekturanalysen zielen darauf ab, potenzielle Risiken (wegen umfangreicher Änderungen) für die Softwarearchitektur zu finden und zu verifizieren, dass die Qualitätsmerkmale in der Softwarearchitektur berücksichtigt wurden [DN02]. Mit Softwarearchitekturanalysemethoden kann schon frühzeitig, bevor die Software (fertig) entwickelt wurde, erkannt werden, ob die Usability in ausreichendem Maße architektonisch unterstützt wird.

Im Rahmen dieser Arbeit wird ein neues Vorgehensmodell zur szenario-basierten Softwarearchitekturanalysemethode erstellt und erprobt, mit dem Softwarearchitekten analysieren können, wie die Usability einer mobilen Anwendung architektonisch unterstützt wird.

1.2 Architektonische Unterstützung von Usability

Die Usability einer Anwendung ist maßgeblich abhängig von der Informationsarchitektur (IA), dem User Interface (UI) und den Nutzer-System-Interaktionen. Die Informationsarchitektur bestimmt, welche Informationen an welchem Ort und auf welche Art und Weise präsentiert werden [RM02]. Das User Interface umfasst die Darstellung der Designelemente und ihres Layouts. Das Interaktionsdesign fokussiert Interaktionen, also jede zielgerichtete Art von Kommunikation zwischen Mensch und Maschine [DFAB03]. Die folgenden Punkte zeigen kurz auf, wie diese Bereiche architektonisch unterstützt werden können.

- Informationsarchitektur: Um beispielsweise für mobile Webanwendungen, die auf verschiedenen Endgeräten laufen sollen, Inhalte automatisch für verschiedene Ausgabekanäle anzupassen, eignen sich Architekturmuster zur Content Adaptation (am Server oder am Proxy [BG07], am mobilen Endgerät [BG10b]). Diese Muster erfordern die Separation und Strukturierung der Informationen (XML) und Bilder (IA), Stildefinitionen (XSL, CSS) für verschiedene Bildschirme und Ausdrucke (UI) sowie architektonische Komponenten, die alle Informationen für verschiedene Ausgabekanäle anpassen.

- User Interface: Um verschiedene Ansichten nicht aufwendig manuell pflegen zu müssen, ist es sinnvoll, die Benutzerschnittstelle vom funktionalen Kern und der Datenbasis zu trennen. Ein Beispiel für diese Separation sind Architekturmuster wie Model-View-Controller und Presentation-Abstraction-Control [BMR+96].

- Interaktionsdesign: Damit die Nutzer zum Beispiel über den Systemstatus informiert werden können, kann das Architekturmuster Blackboard [BMR+96] verwendet werden, bei dem Komponenten einer Anwendung einer zentralen Komponente im Vorfeld festgelegte Daten über den Systemstatus senden, die im UI weiterverarbeitet werden. Eine andere Interaktion ist die Validation von Formulardaten, die verteilt oder lokal auf dem mobilen Endgerät erfolgen kann.

Da mobile Anwendungen in verschiedenen Situationen und an verschiedenen Orten genutzt werden, spielen kontextsensitive Interaktionen eine wichtige Rolle: hier ist der Kontext zu definieren (Modelle über Nutzer, System, Aufgaben [BCK03]), zu überwachen und das System muss schließlich auf Änderungen reagieren [RPM00]. Weitere Beispiele sind Funktionen wie das Rückgängigmachen [FvWB06] oder das Abbrechen von Interaktionen [GJB05].

Schließlich bestehen zwischen Usability und anderen Softwarequalitätsmerkmalen (SQM) Wechselwirkungen (u. a. [Int99, BCK03]). Zum Beispiel führen erhöhte Sicherheitsvorkehrungen oft zu Mehrfacheingaben von Passwörtern, behindern effizientes Arbeiten und provozieren Eingabefehler; Modifizierbarkeit wiederum steht oftmals im Einklang mit Usability (siehe Punkt Informationsarchitektur); ein System darf auf Benutzereingaben nicht zu langsam, aber auch nicht zu schnell reagieren [BJK01].

1.3 FRÜHERE ARBEITEN

Bekannte Methoden, mit denen die architektonische Unterstützung für Usability analysiert werden kann, sind die Methoden „Scenario-based Architecture-Level UsabiliTy Analysis“ (SALUTA) von Folmer et al. [FvGB04] und „Architecture Tradeoff Analysis Method“ (ATAM) [CKK02] von Bass und Kollegen. Beide Methoden sind szenario-basierte SAA.

Mit SALUTA von Folmer et al. [Fol05] wird in vier Schritten bewertet, wie die Softwarearchitektur Usability beeinflusst. Diese Methode wird von einem Analysten und einem Architekten durchgeführt. (1) Nachdem Szenarios erstellt werden, (2) wird die Softwarearchitektur beschrieben. Dabei wird überprüft, inwiefern die komplette Wissensbasis aus Lösungsmustern („Usability-Patterns“) und Anforderungen („Usability-Properties“) in der vorliegenden Softwarearchitektur enthalten sind. (3) Daraufhin erfolgt ein Mapping von Szenarios auf die Usability-Patterns und auf die Usability-Properties. Ausgehend davon diskutieren die Teilnehmer der Analyse, wie gut die einzelnen Szenarios unterstützt werden. Dabei wird die Aussage, dass ein Szenario unterstützt wird, bewertet mit: stark abgelehnt (strongly rejected, - -), abgelehnt (rejected, -), mäßig akzeptiert (medium accepted, -/+), schwach akzeptiert (weakly accepted, +) oder stark akzeptiert (strongly accepted, + +). (4) Schließlich erfolgt eine Aussage darüber, ob die Softwarearchitektur prinzipiell Usability-Attribute und damit die Benutzbarkeit unterstützt.

ATAM [CKK02, BCK03, KKLN05] ist eine umfassende SAA, die viele Qualitätsmerkmale thematisiert, darunter Usability. Ziel dieser Methode ist es, Austauschbeziehungen zwischen Qualitätsmerkmalen und solche Entscheidungen zu ermitteln, die komplexe Änderungen erfordern („Risiken“). Durchgeführt wird diese Methode zuerst von einem Analystenteam und Architekten (Phase 1) und danach von allen Stakeholdern (Phase 2).

Diese Methode beginnt (Schritte 1 bis 4) mit der Betrachtung des Geschäftskontexts und der Diskussion der aktuellen Architektur. (5) Danach wird ein „Qualitätsbaum“ mit anvisierten Qualitätsmerkmalen des Systems erstellt. Anhand dessen sammeln die Teilnehmer in einem Brainstorming Ideen für Szenarios und wählen sie in Abstimmungen aus. (6) Bei der Evaluation der Szenarios stellen Experten mit einem fundierten Wissen und Erfahrung im Bereich Softwarearchitektur Fragen an die Architekten. Die Wissensbasis aus Patterns und Checklisten unterstützt sie und die Stakeholder dabei. (7) In einer zweiten Phase erstellen Stakeholder weitere Szenarios. (8) Gemeinsam werden dann weitere Architekturentscheidungen ermittelt und bewertet. (9) Die Methode nennt abschließend thematisch zusammengefasste Risiken für die Architektur.

1.4 Probleme der früheren Arbeiten

Bezüglich der eigentlichen Analyse sind beide früheren Methoden sehr stark von dem nicht greifbaren Wissen der Anwender der Methode abhängig. ATAM hängt ab von nicht näher bestimmten „Fragen durch Experten"' und bei SALUTA ist es notwendig, existierende Usability-Properties und Usability-Patterns in einer vorliegenden Architektur aufzufinden und später zu bewerten, wie gut sie Usability fördern oder ob sie diese behindern. Beide Vorgehensweisen erfordern es, dass sich Architekten sehr viel Wissen über Architekturmuster bzw. über den Zusammenhang zwischen Usability und Softwarearchitektur aneignen. Dies stellt hohe Einstiegshürden für eine Softwarearchitekturanalysemethode dar.

Außerdem erfordert es die interdisziplinäre Arbeit, die Begriffe und Konzepte eines Qualitätsmerkmals fachübergreifend einheitlich und korrekt zu verwenden, um Missverständnisse zu vermeiden. Usability wird durch Usability-Attribute näher beschrieben (Überblick Tabelle 7, 35). Es ist sinnvoll, eine der Definitionen aus dem Fachgebiet zu verwenden und auch die damit zusammenhängenden Konzepte zu berücksichtigen. In SALUTA werden Usability-Attribute allerdings je nach Nutzergruppe in eine Rangfolge gebracht, um zu verdeutlichen, in welcher Reihenfolge die Usability-Attribute für eine bestimmte Benutzergruppe wichtig sind. Usability-Attribute sind aber nominale Attribute, die Umwandlung in eine Ordinalskala ist unzulässig. In ATAM werden Usability-Attribute verwendet, um davon Szenarios abzuleiten. Im Usability Engineering basieren Anforderungen (u.a. Aufgaben und ableitbare Szenarios) auf dem Nutzungskontext, denn dieser bestimmt laut Usability-Definition die Usability; er umfasst Eigenschaften von Nutzern, Aufgaben, Endgeräten und der Umgebung [PRS07, FJ06]. Es ist daher sinnvoll, Szenarios, wie im Usability Engineering üblich, vom Nutzungskontext abzuleiten [Deu11, FJM05].

Die Aussagekraft beider Methoden erreicht schließlich nicht das Ziel, zu erkennen, inwiefern konkrete Interaktionen architektonisch unterstützt oder behindert werden. Bei ATAM werden zwar Änderungsaufwände für die Softwarearchitektur ermittelt, allerdings geht der Bezug zu konkreten Szenarios verloren [KKLN05]. Mit SALUTA wird ermittelt, ob einzelne Szenarios architektonisch mit bestimmten Usability-Patterns und Usability-Properties unterstützt werden [FvGB04, FB05]; es ist also möglich, zu erkennen, ob bestimmte Lösungen in der Architektur berücksichtigt wurden, aber nicht, inwiefern das genau geschieht.

1.5 Vorgehen in diesem Forschungsprojekt

Diese Arbeit ist, wissenschaftstheoretisch gesehen, in die Kritische Theorie einzuordnen. Kritische Theorie sieht Forschung als politische Aktivität zur Stärkung von Individuen und Gruppen durch Erarbeitung von Wissen. Die Auswahl der Forschungsmethoden ist darauf basiert, wem die Ergebnisse nützen; daher werden partizipatorische Ansätze bevorzugt. [ESSD07]

Typische Forschungsmethoden der Method Construction sind Fallstudien und Action Research, untermauert durch Literaturstudien [BWHW05]. Szenario-basierte Softwarearchitekturanalysemethoden werden zudem generell mit Fallstudien validiert [CKK02, BG04, BLBV04].

Action Research (AR, Anwendungsnahe Forschung) ist ein partizipatorischer Ansatz, an dem die Personen teilnehmen, denen die Forschung etwas nützt. Eine Schwäche von Action Research ist allerdings, dass es nicht rigoros genug erscheint [DMK04, ESSD07]. Deshalb wurde das Canonical Action Research (CAR, die kanonische anwendungsnahe Forschung) aufgesetzt [DMK04]. Kanonische anwendungsnahe Forschung basiert auf fünf Forschungsprinzipien: Vereinbarungen über die Zusammenarbeit zwischen Forschung und Praxis werden explizit getroffen; ein zyklischer Prozess aus Diagnose, Pla-

nung, Intervention (Durchführung), Beurteilung und Reflexion wird eingehalten; theoretische Hintergründe werden offengelegt; Änderungen werden gemeinsam mit den Teilnehmern der Fallstudien beschlossen; Präsentationen und wissenschaftliche Publikationen sollen zudem für alle Beteiligten und die wissenschaftliche Gemeinschaft das Lernen durch Reflexion ermöglichen. [DMK04]

Im Rahmen dieser Forschungsarbeit wurden daher zuerst die theoretischen Grundlagen mittels Literaturstudien erarbeitet; danach wurde eine erste Version der Methode basierend auf Literaturstudien konstruiert. Anschließend wurde der CAR-Prozess zweimal durchlaufen, um die Methode zu erproben und zu verbessern. Die Fallstudien sind (1) die Analyse eines wissenschaftlichen Prototyps für innovative Interaktionsmethoden mit mobilen Endgeräten und (2) die Analyse einer am Markt etablierten Anwendung für Staumeldungen.

1.6 DIE METHODE SATURN

In der Methode „SoftwareArchiTekturanalyse von Usability-anfoRderungeN“ (SATURN) (Kap. 4, S. 48) wird die Softwarearchitektur analysiert, um Architekturentscheidungen (AE, Entscheidungen) zu ermitteln, welche bestimmte Nutzer-System-Interaktionen einer Anwendung unterstützen, be- oder verhindern (szenario-beeinflussende Entscheidungen, S). Die Nutzer-System-Interaktionen sind als Interaktionsszenarios beschrieben. Diese Methode wird im iterativen Prozess der Erstellung der Softwarearchitektur durchgeführt und ist auf mobile Anwendungen spezialisiert.

Die Hauptaktivitäten bei der Anwendung von SATURN sind die Phasen: (1) Beschreiben des Analysekontexts, (2) Bestimmen der Interaktionsszenarios, (3) Evaluieren der Interaktionsszenarios, (4) Interpretieren der Resultate, (5) Retrospektive.

(1) Beschreiben des Analysekontexts (Abschnitt 4.2.3, S. 68ff.): Grob wirtschaftlichen Kontext und Nutzungskontext bestimmen; Interaktionen vom Nutzungskontext ableiten;

(2) Bestimmen der Interaktionsszenarios (Abschnitt 4.2.4, S. 68ff.): Interaktionsszenarios aus dem bereitgestellten Katalog auswählen; zu evaluierende Interaktionsszenarios bestimmen;

(3) Evaluieren der Interaktionsszenarios (Abschnitt 4.2.5, S. 69ff.): für jedes Interaktionsszenario getroffene Architekturentscheidungen ermitteln und evaluieren, inwiefern sie zum jeweiligen Interaktionsszenario konform sind; (optional: falls sie das nicht sind, alternative Architekturentscheidungen brainstormen);

(4) Interpretieren der Resultate (Abschnitt 4.2.6, S. 71ff.): Ergebnisse der Phase 3 zusammenfassen, übergreifende Themen ermitteln und schematisch diskutieren;

(5) Retrospektive (Abschnitt 4.2.7, S. 73ff.): über das Vorgehen und die Ergebnisse der Methode, mit dem Ziel, die Methode und die Interaktionsszenarios kontinuierlich zu verbessern.

Um möglichst viele Usability-Probleme und deren Ursachen zu finden, ist es sinnvoll, eine Softwarearchitekturanalyse mit einer Evaluation der Benutzung selbst zu kombinieren. Deshalb wurde eine Methode über die Koordination von SATURN mit einer Nutzer-Evaluation ergänzt (siehe Kapitel 4 Abschnitt 4.3, S. 74ff.).

Bereitgestellt werden zudem 50 Interaktionsszenarios, die auf existierenden Mustersammlungen basieren, typische Interaktionen mit Anwendungen beschreiben und zu Architekturänderungen führen können, wenn sie in der Architektur noch nicht berücksichtigt wurden. Gültige Interaktionsszenarios basieren auf existierenden Architektur-

bzw. Interaktionsdesign-Patterns. Somit ist es bei SATURN nicht notwendig, alle Patterns zu kennen, die es gibt, denn die relevanten werden referenziert. Durch die Lebenszyklen für Interaktionsszenarios und für Patterns können eigene Interaktionsszenarios und eigene Patterns ergänzt werden.

1.7 Verbesserungen durch die Methode SATURN

Anforderungskontext In der neuen Methode werden an Stelle von Qualitätsattributen von Anfang an konkrete Usability-Anforderungen untersucht. Analog zum Vorgehen im Usability-Engineering werden solche Interaktionsszenarios ermittelt, ausgewählt und bewertet, die entsprechend des Nutzungskontexts der konkreten mobilen Anwendung fachlich relevant sind.

Koordination mit dem Usability Engineering Verschiedene Begriffe, Arbeitsweisen und Sichtweisen behindern bisher die Kooperation zwischen Softwaretechnik und Usability Engineering [FJM05]. Die neue Methode nutzt Usability Engineering-Begriffe und -Techniken, insbesondere die Anforderungserhebung aus dem Nutzungskontext. Wie die neue Methode mit einer Usability-Evaluation kombiniert werden kann, wurde ebenfalls erforscht und erprobt.

Interaktionsszenarios Die Anwender der Methode erhalten einen Katalog von Interaktionsszenarios, die sie für die szenario-basierte Analyse verwenden können. Diese beschreiben Usability-Anforderungen in natürlicher Sprache. Ein gültiges Interaktionsszenario ist aus Patterns extrahiert und stellt typische Anforderungen aus dem Interaktionsdesign dar, die architektursensitiv sein können.

1.8 Überblick über die Kapitel

1. Kapitel „Einleitung“: motiviert, leitet ins Themengebiet ein, fasst frühere Arbeiten zusammen, stellt Probleme vor, erklärt, wie welche Ergebnisse der Forschungsarbeit erreicht werden.

2. Kapitel „Frühere Arbeiten“ (S. 8 ff.): beschreibt Qualitätsziele von Softwarearchitekturanalysemethoden, beschreibt und bewertet die früheren Arbeiten ATAM und SALUTA und stellt darauf basierend Forschungsfragen für diese Arbeit auf.

3. Kapitel „Interaktionsszenarios für die Methode SATURN“ (S. 27 ff.): betrachtet verwandte Arbeiten zu Szenarios für Softwarearchitekturanalysemethoden, definiert Interaktionsszenarios und beschreibt, wie sie erhoben, klassifiziert und gepflegt werden.

   Veröffentlichungen: Mobile Multimedia 2008 [BGG08], Journal it - Information Technology 2009 [GBG09]; Mobile Web Information Systems 2011 [BSG11]

4. Kapitel „Die Methode SATURN“ (S. 48 ff.): thematisiert, wie die Forschungsfragen beantwortet werden, um die Methode SATURN zu erstellen und beschreibt diese Methode danach. Zusätzlich wird eine Methode vorgestellt, mittels welcher diese Softwarearchitekturanalyse mit einem Nutzertest kombiniert werden kann.

   Veröffentlichungen: European Conference on Software Architecture 2009 [BG09], Journal of Systems and Software 2010 [BGG10], Software Engineering and Advanced Applications 2010 [BG10a]

5. Kapitel „Fallstudien“ (S. 84 ff.): erläutert das Studiendesign, dokumentiert die Durchführung und daraus folgende Erkenntnisse.

   Veröffentlichungen: European Conference on Software Architecture 2009 [BG09], Journal of Systems and Software 2010 [BGG10]

6. Kapitel „Validation der Methode SATURN“ (S. 167 ff.): bestimmt Validationsziele bezüglich der Machbarkeit der Methode SATURN und untersucht die theoretischen Grundlagen, den inneren Aufbau der Methode und ihre Ergebnisse auf konstruktive, interne und externe Validität. Thematisiert wird abschließend die Nützlichkeit der neuen Methode im Vergleich mit den früheren Arbeiten.

7. Kapitel „Zusammenfassung“ (S. 199 ff.): fasst die Beiträge der Arbeit abschließend zusammen und skizziert zukünftige Forschungsthemen.

A Anhang (S. 203 ff.): enthält 50 Interaktionsszenarios.

2 FRÜHERE ARBEITEN

Der erste Teil führt ausgehend von den Zielen (Abschnitt 2.1) und allgemeinen verwandten Arbeiten zu den speziell auf Usability bezogenen Softwarearchitekturanalysemethoden (Abschnitt 2.2). Basierend auf den Qualitätsanforderungen an eine Methode zur Analyse der architektonischen Unterstützung von Usability (Abschnitt 2.3) folgen Beschreibungen der früheren Methoden: SALUTA (Abschnitt 2.4) und ATAM (Abschnitt 2.5). Bewertet werden sie anhand der genannten Qualitätsanforderungen in Abschnitt 2.6. Die Forschungsfragen ergeben sich aus den Fragen, die durch die früheren Methoden unbeantwortet geblieben sind (Abschnitt 2.7). Abschnitt 2.8 beschreibt schließlich, wie diese Forschungsfragen in den darauf folgenden Kapiteln behandelt werden.

2.1 SOFTWAREARCHITEKTURANALYSEMETHODEN

Mit Softwarearchitekturanalysen sollen die Wahrscheinlichkeit von Risiken verringert, Qualitätsanforderungen verifiziert und Abhängigkeiten zwischen Architekturentscheidungen verstanden werden [PS15]. Vor dem Hintergrund komplexer Systeme und dem Wunsch nach Wiederverwendung sollen abstrakte Systembeschreibungen erstellt werden, um ein System zu verwalten und zu verstehen [CKK02], insbesondere den Einfluss von Architekturentscheidungen [Kru04] auf Anforderungen [SAR02].

Je nachdem, wann Architekturanalysen stattfinden, werden mit ihnen unterschiedliche Ziele verfolgt: in frühen Stadien des Software-Designs können Stärken und Schwächen von architektonischen Alternativen im Hinblick auf Qualitätsanforderungen verglichen werden [PS15]. In [SAR02] werden allgemeine Ziele genannt und wie folgt Aktivitäten im Lebenszyklus einer Software zugeordnet:

- Rahmen definieren: abstrakte Anforderungen der Domäne und technologische Machbarkeit in der Domäne prüfen,
- Beginn: detaillierte Anforderungen für Klassen von Systemen in der Domäne prüfen und maßgebliche architektonische Prinzipien identifizieren,
- Ausarbeitung: prüfen, ob Anforderungen architektonisch berücksichtigt wurden,
- Konstruktion: allgemeine Überprüfung des Designs und inwiefern es mit der Architektur übereinstimmt,
- Deployment: laufendes System prüfen,
- Evolution: alle genannten Ziele.

Die unterschiedlichen Arten von Softwarearchitekturanalysemethoden unterscheiden Abowd et al. [ABC+96] und Bass et al. [BCK03] in Fragetechniken und Messtechniken. Fragetechniken umfassen szenario-basierte Analysen, Befragungen und Checklisten, um qualitative Fragen zu generieren; zu Messtechniken gehören Metriken, Simulationen, Prototypen und Experimente. Um die jeweiligen Nachteile auszugleichen, ist es sinnvoll, qualitative und quantitative Techniken zu verbinden. Bekannte Evaluationsmethoden gehören zu der einen oder der anderen Kategorie; auch wenn es sinnvoll ist, qualitative und quantitative Techniken zu verbinden, damit die jeweiligen Nachteile ausgeglichen werden können.

Avritzer und Weyuker [AW99] klassifizieren Softwarearchitekturanalysemethoden in erfahrungs-basierte, simulations-basierte und szenario-basierte Methoden sowie

mathematische Modellierung. Erfahrungs-basierte Methoden bauen auf Kenntnissen der Personen auf, die eine Methode durchführen; bei simulations-basierten Evaluationen werden Softwarekomponenten auf einer hohen Abstraktionsebene implementiert; die mathematische Modellierung prüft mit mathematischen Beweisen und Methoden Qualitätsmerkmale wie Verlässlichkeit und Performanz; szenario-basierte Methoden untersuchen eine Softwarearchitektur hinsichtlich bestimmter Qualitätsattribute mittels Szenarios [PS15]. Nach Patidar und Suman sind gerade die szenario-basierten Softwarearchitekturanalysemethoden ausgereift und am weitesten in der Praxis vertreten [PS15].

Ein Szenario ist ein Instrument, um Qualitätsziele zu erheben. Es ist einfach zu erstellen sowie zu verstehen und aufgrund geringer Trainingsaufwände günstig anzuwenden. Szenarios sind in vielen Software- und Systemanalysemethoden wegen ihrer Einfachheit und Effizienz etabliert, beispielsweise in den Fachgebieten User-Interface-Engineering, Usability-Engineering, der Anforderungserhebung, der Performanzmodellierung oder bei Sicherheitsinspektionen. Szenarios sind ein Mittel dazu, vage Qualitäten wie Modifizierbarkeit, Usability, Sicherheit, Fehlertoleranz in eine konkrete und gut verständliche Anforderung zu übersetzen. [CKK02]

2.2 Szenario-basierte Softwarearchitekturanalysemethoden

Um Architekturanalysen zu vergleichen, untersuchten [DN02], [BZJ04] und [PS15] bestehende szenario-basierte Analysemethoden. Dazu zählen die in [PS15] vorgestellten Methoden „Scenario-based Architecture Analysis Method“ SAAM [KABC96], „Architecture Trade-off analysis Method“ ATAM [KKB+98], „Architecture Level Modifiability Analysis“ ALMA [BLBV04], „SAAM for complex scenarios“ SAAMCS [LRV99], „Scenario-based Architecture Reengineering“ SBAR [BB98], „Architecture Level Prediction of Software Maintenance“ ALPSM [BB99], „Extending SAAM by Integration in the domain“ ESAAMI [Mol99] und die „Software Architecture Comparison Method“ SACAM [BFJK99].

Bei der „Scenario-based Architecture Analysis Method“ SAAM [KABC96] wird eine Architektur in sechs teilweise parallelen Aktivitäten hinsichtlich Modifizierbarkeit und potenziell hohen Änderungsaufwänden („Risiken“) untersucht. Mit Szenarios werden Funktionen und potenzielle Änderungen überprüft. Angewandt wird die Methode, sobald Funktionen bestimmten Modulen zugeordnet sind. Alle Stakeholder nehmen teil; als Werkzeug ist teilweise SAAMTOOL [Kaz96] einsetzbar.

Die „Architecture Trade-off analysis Method“ ATAM [KKB+98] prüft in sechs Aktivitäten und zwei Phasen mehrere Qualitätsmerkmale, darunter u.a. Verfügbarkeit, Sicherheit und Usability. Sie zielt darauf ab, potenzielle Risiken und Austauschbeziehungen zwischen untersuchten Qualitätsmerkmalen aufzudecken. Mit Frage- und Messtechniken (je nach Qualitätsattribut) wird eine Architektur nach der detaillierten Ausarbeitung oder in einem iterativen Verbesserungsprozess (ohne Werkzeugunterstützung) untersucht. Alle relevanten Stakeholder nehmen teil.

Die „Architecture Level Modifiability Analysis“ ALMA [BLBV04] fokussiert in fünf Aktivitäten darauf, Wartbarkeitskosten zu schätzen, Risiken zu untersuchen bzw. Architekturen auszuwählen. Evaluationsmethoden hängen vom Ziel einer Analyse ab. Angewendet wird ALMA während des Designs. Es nehmen nur bestimmte Stakeholder an bestimmten Aktivitäten teil (keine Werkzeugunterstützung).

„SAAM for complex scenarios“ SAAMCS [LRV99] prüft die Architektur mit drei Aktivitäten (zwei verlaufen parallel) hinsichtlich Flexibilität und zielt auf das Erkennen von Risiken ab. Es werden komplexe Szenarios erstellt, mit denen eine finale Architektur mit allen Stakeholdern untersucht wird (keine Werkzeugunterstützung).

Die Methode „Scenario-based Architecture Reengineering“ SBAR [BB98] erhebt mit drei Aktivitäten, wie verschiedene Qualitätsmerkmale unterstützt werden. Ziel ist es,

eine Architektur zu überarbeiten. Dabei nutzt ein Architekt verschiedene Evaluationsansätze, um eine Architektur (ohne Werkzeugunterstützung) umzuarbeiten bzw. zu erweitern.

Die „Architecture Level Prediction of Software Maintenance“ ALPSM [BB99] fokussiert sich in sechs Aktivitäten darauf, die Wartbarkeit einer Software vorherzusagen. Die Methode nutzt eine szenario-basierte Analyse durch einen Softwarearchitekten während der Erstellung der Architektur (ohne Werkzeugunterstützung).

„Extending SAAM by Integration in the domain“ ESAAMI [Mol99] geht analog zu SAAM vor, analysiert also eine Architektur bezüglich Modifizierbarkeit, verwendet aber eine wiederverwendbare Wissensbasis und fokussiert sich auf eine bestimmte Domäne. Die Methode wird von allen Stakeholdern mit der finalen Version der Softwarearchitektur und ohne Werkzeugunterstützung durchgeführt.

Die „Software Architecture Comparison Method“ SACAM [BFJK99] fokussiert verschiedene Qualitätsattribute. In sechs Aktivitäten (eine wird wiederholt) vergleichen Softwarearchitekten ohne Werkzeugunterstützung die Architekturen unterschiedlicher Domänen.

Alle diese Methoden basieren auf SAAM. Am weitesten ausgereift sind die „Architecture Trade-off analysis Method“ ATAM [KKB+98] und die „Architecture Level Modifiability Analysis“ ALMA [BLBV04]. [PS15]

Vilela et al. [VFCS15] untersuchten in einer Literaturstudie den aktuellen Stand der Forschung zum Thema Usability und Softwarearchitektur. Neben dem Workshop für Usability-Szenarios [RRD06, RRD07] (siehe Abschnitt 3.2.2, S. 30) für SAAM bzw. ATAM wurden SATURN und SALUTA als Architekturanalysemethoden vorgestellt. SALUTA steht für „Scenario-based Architecture-Level UsabiliTy Analysis“ und basiert auf ALMA [Folo5]. ATAM ist die Nachfolgemethode von SAAM und ist noch deutlicher auf Usability ausgerichtet: sie verwendet Usability-Taktiken und die genannten Usability-Szenarios [CKK02, BCK03]. Die stark auf Usability ausgerichteten Methoden ATAM und SALUTA werden deshalb genauer betrachtet.

2.3 Qualitätsanforderungen an szenario-basierte Softwarearchitekturanalysemethoden

Um szenario-basierte Softwarearchitekturanalysemethoden inhaltlich zu vergleichen, eignen sich die von Kazman und Kollegen [KKLN05] beschriebenen vier Qualitätsanforderungen an solche Methoden.

1. Bestimmung von Kontext und Qualitätsmerkmalen: Kontext und Qualitätsmerkmale sollen klar definiert sein und strukturiert erfasst werden. Falls die Softwarearchitektur einen bestimmten Entwicklungsstatus haben muss, damit die Analyse durchgeführt werden kann, soll dieser definiert sein und vor Beginn der Softwarearchitekturanalysemethode überprüft werden.

2. Bestimmung und Selektion von Szenarios: Die Szenarios sollen so definiert sein, dass sie die Qualitätsmerkmale sinnvoll operationalisieren. Ihre Beschreibung, Priorisierung und anschließende Auswahl soll sinnvoll und nachvollziehbar sein. Es soll außerdem angeleitet werden, wie die Teilnehmer mit nicht-technischen Themen umgehen, die während der Methode aufkommen.

3. Unterstützung der Analyse: Hintergrundmaterial, Vorlagen und Anleitung sollen standardisiert, verständlich und ausführlich sein.

4. Bestimmen der Analyseergebnisse: Von den Qualitätsmerkmalen ausgehend soll die Softwarearchitekturanalysemethode nachvollziehbar und wiederholbar zu den jeweiligen Ergebnissen führen. Diese sollen wiederum den betrachteten

Qualitätsmerkmalen zugeordnet werden können. Große Freiheitsgrade sollen bei der Anwendung der Methode vermieden werden.

Im Usability Engineering arbeiten verschiedene Disziplinen zusammen; sie sollten daher miteinander kooperieren. Eine Softwarearchitekturanalysemethode für dieses Fachgebiet sollte also auch Voraussetzungen für eine Kooperation [AP99] schaffen.

5. Zusammenspiel von Prozessen: Grundsätzlich sollen die beteiligten Gruppen kommunizieren, also Informationen austauschen. Um Arbeit zu koordinieren, müssen Kommunikationsprozesse etabliert sein. Kooperation wird schließlich erreicht, wenn arbeitsteilige Aktivitäten aufeinander abgestimmt sind.

In den nächsten Abschnitten werden die betrachteten verwandten Arbeiten beschrieben und bewertet.

2.4 BESCHREIBUNG DER METHODE SALUTA

2.4.1 *Kurzvorstellung der Methode SALUTA*

Mit der Methode „„Scenario-based Architecture-Level UsabiliTy Analysis““ (SALUTA) von Folmer et al. [Fol05] wird analysiert, wie das Qualitätsmerkmal Usability von Softwarearchitektur beeinflusst werden kann. SALUTA wurde von der szenario-basierten Methode „Architecture-Level Modifiability Analysis“ (ALMA) [BLBV04] abgeleitet. Die Schritte der Methode sind: (1) Erstelle Nutzungsprofile, (2) Beschreibe die angebotene Usability, (3) Evaluiere die Szenarios, (4) Interpretiere die Resultate [FB05].

1. In Schritt 1 „Erstelle Nutzungsprofile“ werden einzelne Interaktionen von Systemrollen als Nutzungsprofil („usage profile“) aufgelistet. Damit sollen Usability-Anforderungen beschrieben werden, die zu guter Usability führen: „describe required usability.“ [FB05]

2. In Schritt 2 „Beschreibe die angebotene Usability“ werden Informationen über die Softwarearchitektur ermittelt [FB05]. Die Architekturdokumentation wird durchgesehen und Architekten werden befragt, ob und wie die Usability-Properties und Usability-Patterns der Wissensbasis in der Architektur berücksichtigt wurden [FvGB04].

3. In Schritt 3 „Evaluiere die Szenarios“ wird bestimmt, ob die Szenarios des Nutzungsprofils durch die Softwarearchitektur unterstützt werden können. [FB05]

4. In Schritt 4 „Interpretiere die Resultate“ werden die Ergebnisse gedeutet und Aktivitäten abgeleitet. [FB05]

Usability wird in SALUTA über Usability-Attribute definiert, wobei verschiedene Usability-Definitionen gegenübergestellt und ähnliche Usability-Attribute zusammengefasst wurden unter: Erlernbarkeit, Effizienz bei der Benutzung, Verlässlichkeit bei der Benutzung und Zufriedenheit [FGB03].

Der Nutzungskontext, in dem sich Nutzer einer Anwendung immer befinden, wird betrachtet und mit einem Wort beschrieben, z. B. mobil, Desktop [FvGB04].

Nach dieser Kurzvorstellung zu Struktur, Qualitätsmerkmal und Kontext soll nun das Vorgehen genauer betrachtet werden.

2.4.2 *Detailliertes Vorgehen in der Methode SALUTA*

Im ersten Schritt werden Szenarios erstellt, d.h. 3-Tupel (user, task, context of use), die benutzerinitiierte (und keine vom System initiierten) Interaktionen beschreiben. Ein Beispiel ist (Admin, Login, Desktop). In der Fallstudie Webplatform [FvGB04] wurden gemeinsam mit einem Usability Engineer 30 Richtlinien nach Nielsen [Nie93] in Szenarios umgewandelt, darunter die Anforderung „every page should feature a quick search which searches the whole portal and comes up with accurate search results"; die umgewandelte Anforderung lautet „quick search". Die Suche soll schnell sein, sich auf das ganze Portal beziehen und korrekte Ergebnisse liefern. Das reduzierte Szenario enthält allerdings nur die erste Information.

Für jede Nutzergruppe werden eigene Szenarios erstellt. So wurden bei der Fallstudie Webplatform (Portal für eine Universitäts-Website) vier Interaktionen in elf Szenarios überführt: je dreimal „Quick Search", „Navigate", „Edit Object" und zweimal „Make Object" [FvGB04]. Bei der Fallstudie eSuite (Portal für Enterprise Resource Planning) wurden fünf Interaktionen in zwölf Szenarios überführt: „Insert Order" (viermal) und je zweimal „Search Order", „Get Balance", „View Stock", „Get Statistics" [FB05].

Um darzustellen, wie stark ein Szenario die Usability voraussichtlich beeinflusst, werden Usability-Attribute in eine Rangfolge (Ordinalskala) gebracht. Ob ein Szenario für eine Benutzergruppe besonders wichtig ist, wird indirekt durch die Reihenfolge der Usability-Attribute abgebildet (siehe Abbildung 1 aus der Fallstudie zur Web-Anwendung eSuite). Die Tabelle beschreibt in der ersten Zeile im Szenario 1: „Einfache Nutzer" führen im Kontext „Desktop" die Aufgabe „Auftrag eingeben" durch. Usability-Attribute wurden geordnet: Erlernbarkeit, Verlässlichkeit bei der Benutzung, Zufriedenheit und Effizienz. Zeile drei betrifft die gleiche Interaktion, aber die Nutzer „Administratoren". Die Reihenfolge Usability-Attribute ist: Effizienz, Verlässlichkeit bei der Benutzung, Zufriedenheit und Erlernbarkeit. [FB05]

#	User	Context	Task	e	l	r	s
1	Simple users	Desktop	Insert Order	1	4	3	2
2	Simple users	Mobile	Insert Order	4	3	2	1
3	Administrators	Desktop	Insert Order	4	1	3	2
4	Administrators	Mobile	Insert Order	4	1	3	2
5	Simple users	Desktop	Search Order	1	3	4	2
6	Administrators	Desktop	Search Order	3	1	4	2
7	Simple users	Desktop	Get balance	1	4	3	2
8	Administrators	Desktop	Get balance	3	1	4	2
9	Simple users	Desktop	View Stock	1	4	2	3
10	Administrators	Desktop	View Stock	4	1	2	3
11	Simple users	Desktop	Get Statistics	1	4	2	3
12	Administrators	Desktop	Get Statistics	4	1	3	2

– Nummer, e – Effizienz, l – Erlernbarkeit,
r – Verlässlichkeit bei der Benutzung, s – Zufriedenheit

Abbildung 1: Nutzungsprofil der Fallstudie der Anwendung eSuite [FB05]

Anschließend werden Szenarios für die Analyse selektiert, allerdings ohne Kriterien zu nennen [FvGB04]; im Beispiel wurden die Szenarios 3 (Admins, Desktop, Insert Order), 5 (Simple Users, Desktop, Search Order) und 6 (Admins, Desktop, Search Order) gewählt (siehe Ergebnisdarstellung, Abbildung 4, Seite 14) [FB05].

Im Schritt 2 diskutieren die Analysten darüber, welche Elemente der Wissensbasis in der Softwarearchitektur berücksichtigt wurden. Unterstützt wird die Analyse durch folgendes Hintergrundmaterial.

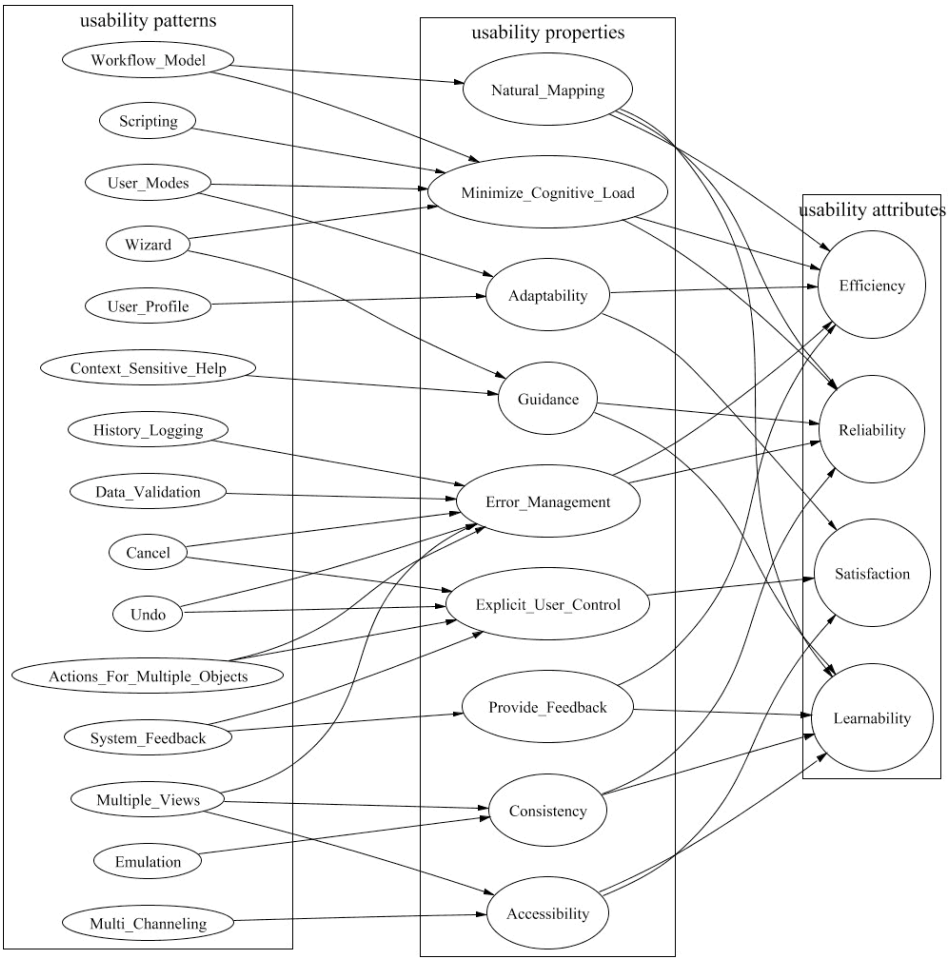

Abbildung 2: Software-Architecture-Usability-Framework [FGB03]

SALUTA stellt das „Software-Architecture-Usability-Framework (SAU-Framework)“ (Abbildung 2) als Wissensbasis bereit [FGB03]. Da es die Szenarios ausschließt, ist es zwar nicht das Metamodell der Methode, aber eine Interpretationshilfe. Es fasst theoretische Zusammenhänge zwischen Usability-Attributen, „Usability-Patterns“ und „Usability-Properties“ zusammen. Usability-Patterns wie CANCEL, UNDO, WIZARD und EMULATION enthalten grob Entwurfsvorschläge; sie resultieren aus einer Literaturstudie zu Best Practices sowie Interviews mit Architekten. Die ebenfalls per Literaturstudie ermittelten Usability-Properties umfassen allgemeine Usability-Heuristiken wie beispielsweise Konsistenz, Minimierung des kognitiven Aufwandes, kontextsensitive Hilfe und Adaptionsfähigkeit. Gerichtete Linien zwischen Usability-Patterns, Usability-Properties und Usability-Attributen deuten auf eine positive oder negative Beeinflussung hin. So wird beispielsweise das Pattern WIZARD eingesetzt, damit Nutzer angeleitet werden (Linie zu Usability-Property „Guidance“), so dass die Benutzung leicht erlernbar und weniger fehleranfällig ist (Linien zu Usability-Attributen „Learnability“ und „Reliability“). Für dieses Framework wird kein Anspruch auf Allgemeingültigkeit erhoben. [FGB03]

Abbildung 3 zeigt eine „Architecture-Support-Matrix“, die im Rahmen der Fallstudie Webplatform erstellt wurde. Das nicht näher bestimmte Szenario 1 wird unterstützt, indem in der Softwarearchitektur (SA) die Usability-Patterns SCRIPTING, MULTIPLE VIEWS, MULTI CHANNELING, EMULATION und CONTEXT SENSITIVE HELP sowie die Usability-Properties „Consistency“, „Guidance“ und „Accessibility“ berücksichtigt wurden. [FvGB04]

Scenario number	Usability patterns															Usability properties									
	System Feedback	Actions for multiple obj.	Cancel	Data validation	History Logging	Scripting	Multiple views	Multi Channeling	Undo	User Modes	User Profiles	Wizard	Workflow model	Emulation	Context sensitive help	Provide feedback	Error management	Consistency	Adaptability	Guidance	Explicit user control	Natural mapping	Accessibility	Minimize cognitive load	Explorability
1	×	×	×	×	×	✓	✓	✓	×	×	×	×	×	✓	✓	×	×	✓	×	✓	×	×	✓	×	×
2	✓	×	×	×	×	✓	✓	✓	×	✓	✓	×	×	✓	✓	✓	×	✓	✓	✓	×	×	✓	×	×
3	✓	×	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	×	×
4	✓	×	×	✓	✓	✓	✓	✓	✓	×	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	✓	×	×

Abbildung 3: Ergebnisse der Fallstudie Webplatform aus [FvGB04]

Im Schritt 3 entscheiden die Analysten, inwiefern der Annahme zugestimmt werden kann, dass ein Szenario unterstützt wird. Antwortmöglichkeiten sind: stark abgelehnt (strongly rejected, - -), abgelehnt (rejected, -), mäßig akzeptiert (medium accepted, -/+), schwach akzeptiert (weakly accepted, +) und stark akzeptiert (strongly accepted, + +). Abbildung 4 zeigt ein Beispiel aus [FB05], in dem drei Analysen zusammengefasst werden. Die alte und die neue Version der Anwendung Compressor (Online-Katalog zum Vergleichen von Kompressoren) und die Anwendung eSuite wurden analysiert. Im Beispiel wurde das Szenario 1 der Anwendung „Compressor“ in der alten Version durch kein Usability-Pattern und keine Usability-Property gefördert (Anzahl der zugeordneten Usability-Patterns bzw. Usability-Properties ist jeweils mit 0 von 15 angegeben). Die Aussage, dass das Szenario unterstützt wird, wird „stark abgelehnt“ („support“ ist „- - “). Analog dazu fasst die dritte Zeile die Ergebnisse der Evaluation der Szenarios zusammen: in der neuen Anwendung wurden 6 der 15 Usability-Patterns und 5 der 10 Usability-Properties integriert. Die Aussage, dass das Szenario 1 in der neuen Version von „Compressor“ unterstützt wird, wird „mäßig akzeptiert“ („support“ ist „-/+“). [FB05]

	Scenario #	**# patterns**	**# properties**	**support**
Compressor	1 (old)	0/15	0/15	--
	2 (old)	1/15	1/15	-/+
	1 (new)	6/15	5/10	-/+
	2 (new)	10/15	5/10	+
eSuite	3	8/15	7/10	+/-
	5	8/15	7/10	+
	6	8/15	7/10	++

Szenario # – Nummer des Szenarios, # patterns – Anzahl der zugeordneten Usability-Patterns,
properties – Anzahl der zugeordneten Usability-Properties,
support – Bewertung der Unterstützung durch die Softwarearchitektur

Abbildung 4: Ergebnisdarstellung in SALUTA, Tabelle aus [FB05]

Wie die Analyse durchlaufen wird, wird wie folgt zusammengefasst: „By analyzing for each pattern and property, the effect on usability, the support for this scenario is determined. Due to the lack of formalized knowledge at the architecture level, this step is very much guided by tacit knowledge (i.e. the undocumented knowledge of experienced software architects and usability engineers).“ Diese Anleitung zeigt, dass undokumentiertes Wissen und Erfahrung für die Analyse maßgeblich sind; damit ist schwierig nachvollziehbar, wie die Analyse tatsächlich durchgeführt wird. Dokumentationsvorgaben gibt es nicht, die Veröffentlichungen bieten allerdings konkrete Beispiele an. [FvGB04, FB05].

Im Schritt 4 werden die Ergebnisse interpretiert. Es ergeben sich Richtungsaussagen darüber, ob die betrachtete Softwarearchitektur prinzipiell durch berücksichtigte Patterns der Wissensbasis eine gute Usability ermöglicht. [FvGB04, FB05]

Im Anschluss an die Methode kann, nachdem die Anwendung erstellt wurde, eine Usability-Evaluation durchgeführt werden. So wurde erwähnt, dass spätere Usability-Tests die Ergebnisse von SALUTA bestätigt haben [FvGB04, FB05].

2.5 BESCHREIBUNG DER METHODE ATAM

2.5.1 *Kurzvorstellung der Methode ATAM*

Mit der „Architecture Tradeoff Analysis Method“ (ATAM) [BCK03, CKK02] werden die architektonische Unterstützung von verschiedenen Qualitätsmerkmalen und deren Austauschbeziehungen analysiert. Das Ziel dieser Methode ist es, potenzielle Probleme für die Softwarearchitektur („Risiken“) zu ermitteln. ATAM ist eine Weiterentwicklung der „Software Architecture Analysis Method“ (SAAM) [KABC96]. Das Vorgehensmodell dieser Methode und Beispielfallstudien von ATAM sind in [CKK02, BCK03] aufgeführt.

Die neun Hauptaktivitäten der Methode enthalten zwei Analysedurchgänge: einen ersten mit Analysten und Architekten („Phase 1“), danach einen zweiten mit allen Stakeholdern („Phase 2“). Inhaltlich sind die Hauptaktivitäten in vier Gruppen unterteilt: Präsentation, Untersuchung und Analyse, Testen und Berichten [CKK02]. Vor Durchführung dieser Methode prüfen Analysten, ob die Softwarearchitektur für die Analyse ausführlich genug beschrieben ist [BCK03].

PHASE 1

- Präsentation
  1. Präsentieren von ATAM
  2. Präsentieren der Geschäftsziele als Haupttreiber der Softwarearchitektur
  3. Präsentieren der Softwarearchitektur unter Berücksichtigung der Geschäftsziele
- Untersuchung und Analyse
  4. Identifizieren der architektonischen Ansätze (Softwarearchitektur-Patterns[1])
  5. Generieren des Qualitätsbaumes

     In der ersten Phase spezifizieren Analysten und Architekten Qualitätsmerkmale in einem Qualitätsbaum („Utility-Tree“). Die Attribute der Qualitätsmerkmale sind die Knoten des Baumes. Von ihnen werden in einem Brainstorming Szenarios abgeleitet. Diese werden den jeweiligen Attributen als Blätter zugeordnet.

     Da meist sehr viele Szenarios ermittelt werden und nicht alle geprüft werden können, werden einige ausgewählt. Es werden Prioritäten vergeben: die Architekten und Analysten bewerten anhand der Präsentationen, wie bedeutend ein Szenario für die Stakeholder ist und schätzen zudem, wie komplex die Änderungen sein können. Ausgewählt werden dann beispielsweise nur die Szenarios, die zu den Qualitätsmerkmalen gehören, die von den Stakeholdern als Hauptziele genannt wurden und voraussichtlich sehr komplexe Änderungen erfordern. [CKK02]

1 Patterns sind im Abschnitt 3.4.2 (S. 40f.) genauer erläutert. Von den zwei Denkschulen hinsichtlich der Softwarearchitektur-Patterns vertritt das SEI die *Architekturstile*. Architekturpatterns und Architekturstile sind in ihrer Essenz die gleichen Konzepte und unterscheiden sich durch verschiedene Beschreibungsformate.

6. Analysieren der architektonischen Ansätze

   Die Analysten befragen die Architekten und prüfen die Softwarearchitekturdokumentation. Die Antworten führen zu „Sensitivity Points“: Architekturentscheidungen (AE) mit einem Effekt auf das Szenario. Als „Risiken“ werden potenziell problematische Entscheidungen beschrieben, die das Szenario verhindern. Es gibt auch „Nicht-Risiken“, die ein Szenario nicht verhindern. Beeinflusst eine sensitive Entscheidung ein oder mehrere andere Qualitätsmerkmale, ist sie ein „Tradeoff Point“. [CKK02]

PHASE 2

- Testen
  7. Brainstorming und Priorisieren von Szenarios

     Die Stakeholder ermitteln nun weitere Szenarios. Auch sie führen dafür ein Brainstorming durch. In einem Abstimmungsprozess werden die Szenarios priorisiert: Jede Person erhält beispielsweise 30 Punkte und kann sie auf ein oder mehrere Szenarios verteilen. Die Szenarios mit den meisten Stimmen werden dann (teilweise erneut) analysiert. [CKK02]
  8. Analysieren der architektonischen Ansätze (Wiederholung)
- Berichten
  9. Präsentieren der Ergebnisse

Die Methode erfordert viele Teilnehmer: Alle Stakeholder des betrachteten Systems und ein Team von drei bis vier Analysten (ATAM-Experten) führen diese Methode durch. [BCK03]

Der Analysekontext wird ermittelt, indem die Geschäftsziele sowie die technischen Rahmenbedingungen der Softwarearchitektur betrachtet werden. Der Nutzungskontext wird nicht separat abgehandelt. Von den Geschäftszielen der jeweiligen Anwendung werden Qualitätsmerkmale abgeleitet, die analysiert werden sollen. Das können beispielsweise Performanz, Sicherheit, Modifizierbarkeit und Usability sein.

Nach dieser Kurzvorstellung zu Struktur, Qualitätsmerkmal und Kontext wird nun das Vorgehen genauer betrachtet.

2.5.2 *Detailliertes Vorgehen in der Methode ATAM*

Bei ATAM bestimmen Softwarearchitekten, wie sie Usability operationalisieren, indem sie den Qualitätsbaum erstellen [BCK03]. Dazu erstellen sie in einem Brainstorming Szenarios. Analysten unterstützen sie dabei und redigieren die Ergebnisse. Die Szenarios werden als „Quality Attribute Scenario“ bezeichnet. Sie bestehen aus sechs Elementen:

1. Quelle des Stimulus: ein System oder eine Person,
2. Stimulus: das Attribut eines Qualitätsmerkmals; Kondition, die beim Eintreffen von dem System berücksichtigt werden muss,
3. Umgebung: Konditionen bzw. Kontext, in dem der Stimulus auftritt,
4. Artefakt: der stimulierte Teil des Systems oder das ganze System,
5. Response: Aktivität nach Ankunft des Stimulus,
6. Response Measure: wie die Anforderung getestet werden kann.

Die Generierungstabelle für „Usability-Szenarios“ (Abbildung 1) aus [BCK03] zeigt, wie die Usability-Szenarios gestaltet sein können. Usability-Attribute und Usability-Anforderungen werden dabei vermischt: Erlernbarkeit, Effizienz und Zufriedenheit sind Usability-Attribute, die Minimierung des Einflusses von Fehlern und die Adaptierbarkeit sind Usability-Anforderungen [BCK03].

Element des Szenarios	*Mögliche Werte*
Quelle	Endbenutzer
Stimulus	Auswahl eines Usability-Attributes als Stimulus: • System-Features kennenlernen (Erlernbarkeit) • das System effizient nutzen (Effizienz) • den Einfluss von Fehlern minimieren (Fehlertoleranz) • das System anpassen (Adaptierfähigkeit) • sich wohl fühlen (Zufriedenheit)
Artefakt	System
Umgebung	Laufzeit oder Konfigurationszeit
Response	Das System reagiert mit einer oder mehreren Responses: • zur Unterstützung von Erlernbarkeit: kontextsensitives Hilfesystem, vertrautes User Interface, in ungewohntem Kontext nutzbares User Interface • zur Unterstützung von Effizienz: Aggregation von Daten und/oder Befehlen, Wiederverwendung von schon eingegebenen Daten und/oder Befehlen, Unterstützung effizienter Navigation, konsistente Befehle in verschiedenen Sichten, umfassende Sichten, Multitasking • zur Unterstützung von Fehlertoleranz: Rückgängigmachen, Abbrechen, Wiederherstellen des letzten Status nach Systemabsturz, Erkennen und Korrigieren von Benutzerfehlern, alternative Zugriffsmöglichkeiten bei vergessenem Passwort, Verifikation von Systemressourcen • zur Unterstützung von Adaptierfähigkeit: Personalisierung, Internationalisierung, automatische Anpassung an Benutzer/die Benutzung • zur Unterstützung von Zufriedenheit: Systemstatus anzeigen, in der Geschwindigkeit des Nutzers arbeiten
Response Measure	Beispiele: Aufgabendauer, Anzahl von Benutzerfehlern, Anzahl gelöster Probleme, Benutzerzufriedenheit, Wissenszuwachs des Benutzers, Anteil erfolgreicher Befehle zu allen Befehlen, Umfang verlorener Daten und/oder verlorener Zeit

Tabelle 1: Generierungstabelle für Usability-Attribut-Szenarios [BCK03]

Ein Beispiel aus dieser Generierungstabelle ist folgendes: „Ein Benutzer (Quelle) will sich während der Benutzung des Systems (Artefakt) zur Laufzeit (Umgebung) wohl fühlen (Stimulus). Der Systemstatus wird angezeigt, und das System arbeitet mit der Geschwindigkeit des Benutzers (Response). Die Benutzerzufriedenheit soll gut sein (Response Measurement).“ Dem Stimulus „sich wohl fühlen“ wurde also die Response zugeordnet „der Systemstatus wird angezeigt“.

Im Vorfeld von ATAM können Stakeholder einen Workshop besuchen, der behandelt, wie sie solche Usability-Szenarios ausgehend von Usability-Attributen ermitteln können [RRD06] (Kapitel 3, Abschnitt 3.2.2, S. 30).

Die Selektion von Szenarios erfolgt im ersten Durchgang durch Fachleute, im zweiten Durchgang mit allen Stakeholdern. Im ersten Durchgang bewerten Architekten und Analysten, inwiefern bekannte Anforderungen der Stakeholder für ein Szenario sprechen und welche Änderungen ein Szenario erfordern könnte. Dabei setzt die letztere Priorisierung voraus, dass der Änderungsaufwand der Softwarearchitektur, den das Szenario verursachen kann, bekannt ist. Das kann aber erst ermittelt werden, wenn die Softwarearchitektur analysiert wird. So können auch Szenarios ausgeschlossen werden, deren Analyse Probleme offenlegen würde. Im zweiten Durchgang stimmen Stakeholder über Szenarios ab. Jeder Teilnehmer kann eine bestimmte Gesamtpunktzahl (aufgerundet 30 Prozent der Anzahl aller Szenarios) auf ein oder mehrere Szenarios verteilen. Nach Bass et al. [BCK03] ist dies eine übliche Priorisierung beim Brainstorming und eine sinnvolle Möglichkeit, bei einer großen Anzahl von Szenarios und Teilnehmern zu einer Einigung zu gelangen. Allerdings ist dieses Vorgehen auch manipulierbar: Wenn ein Stakeholder jeweils die gleiche Anzahl an Punkten auf bestimmte Szenarios verteilt und ein anderer Stakeholder die doppelte Punktzahl auf die Hälfte der gleichen Szenarios, dann zählt nur die Meinung des zweiten Stakeholders. Unabhängig von der fachlichen Relevanz seiner Argumente hat dann der zweite Stakeholder seine Auswahl durchgesetzt.

Die Analyse erfolgt, indem Analysten Fragen an die Architektur stellen: diese Fragen werden nicht bestimmt, sie hängen vom Erfahrungswissen der Experten ab. Es gibt zwei Werkzeuge, welche die Analysten unterstützen: „Taktiken“ und „Usability Supporting Architectural Patterns“ (USAPs). Taktiken nennen stichwortartig grundlegende Lösungen zur Umsetzung von Qualitätsmerkmalen. Die Usability-Taktik wird in Abbildung 5 (S. 18) veranschaulicht. Dieses Werkzeug erfordert Expertise: „Tactics [...] are deficient in two fashions, however. First [...] we have tactics for a limited number of qualities. Secondly, tactics must be used with judgement“ [KKLN05].

Abbildung 5 zeigt die Usability-Taktik aus [BCK03] und stellt grundlegende architektonische Entscheidungen dar, die Usability fördern. Dazu gehören die Separation des User Interfaces, die Unterstützung von nutzer- und von system-initiierten Dialogen.

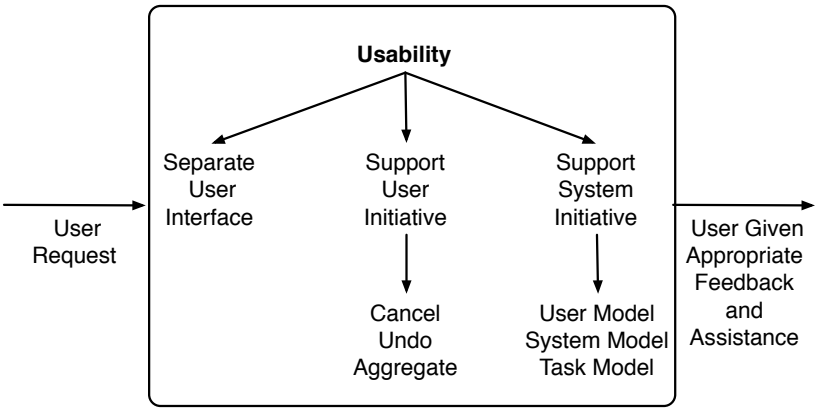

Abbildung 5: Usability-Taktiken [BCK03]

Zusätzlich erforschten Golden et al. [GJB05] den Einsatz von USAPs, ein Beispiel enthält die Abbildung 6 (S. 20). USAPs („Usability Supporting Architectural Patterns“ [GJB05]) basieren auf Architekturpatterns und Erfahrungen [CKK02]. Ein USAP enthält ein Usability-Szenario und eine Liste von allgemeinen Verantwortlichkeiten, die sich vom Nutzungskontext ableiten lassen, sowie eine Beispiellösung.

Die Checklisten werden in einer Webanwendung bereitgestellt [SBJG09]. Die USAPs sind von einer Vielzahl an Patterns abgeleitet und listen zu beachtende Verantwortlichkeiten auf. Sie sind pattern-unabhängig und bieten Informationen mit verschiedenen Detailgraden an, wobei Architekten selbst entscheiden müssen, welche Verantwortlich-

keiten sie priorisieren. Die folgenden Beispiele zum USAP Cancel zeigen, wie detaillierte und allgemein gehaltene Anforderungen vermischt sind [GJB05]:

- „CR1: Ein Button, Menüeintrag, Keyboard-Shortcut oder andere Mittel müssen angeboten werden, mit denen der Nutzer einen aktiven Befehl abbrechen kann."
- „CR2: Das System muss immer den Cancel-Befehl oder Änderungen in Systemumgebung verzeichnen."
- „CR3: Das System muss immer Informationen aufzeichnen (Status, Ressourcenverwendung, Aktionen, etc.), die es ermöglichen, den Systemstatus vor der Ausführung des aktuellen Befehls wieder herzustellen."
- „CR17: Das System muss die zum Abbrechen notwendige Zeit innerhalb von 20% schätzen [es wurde nicht spezifiziert, wovon.]"
- „CR18: Das System muss den Nutzer über diese Schätzung informieren. i. Wenn zwischen 1 und 10 Sekunden geschätzt wurden, muss nur der Cursor geändert werden. ii. Wenn mehr als 10 Sekunden geschätzt werden und die Zeitschätzung bei 20% liegt [es wurde nicht spezifiziert, wovon], ist ein Progress-Indikator besser. iii. Wenn mehr als 10 Sekunden geschätzt werden und kein akkurater Wert festgelegt werden kann, muss eine andere Art von Feedback an die Nutzer übermittelt werden."

Abbildung 6 zeigt die komplette Liste des Usability Supporting Architectural Patterns für Cancel [GJB05].

CR1	A button, menu item, keyboard shortcut and/or other means must be provided, by which the user may cancel the active command.
CR2	The system must always listen for the cancel command or changes in the system environment.
CR3	The system must always gather information (state, resource usage, actions, etc.) that allow for recovery of the state of the system prior to the execution of the current command.
CR4	The system must acknowledge receipt of the cancellation command appropriately within 150 ms. Acknowledgement must be appropriate to the manner in which the command was issued i. For example, if the user pressed a cancel button, changing the color of the button will be seen. ii. If the user used a keyboard shortcut, flashing the menu that contains that command might be appropriate.
CR5	If the command itself is able to cancel itself directly at the time of cancellation, the command must respond by canceling itself (i.e., it must fulfill responsibilities CR9-CR19 below (e.g., an object-oriented system would have a cancel method in each object)).
CR6	If the command itself is not able to cancel itself directly at the time of cancellation, an active portion of the system must ask the infrastructure to cancel the command, or must fulfill responsibility CR7 below.
CR7	If the command itself is not able to cancel itself directly at the time of cancellation, the infrastructure itself must provide a means to request the cancellation of the application (e.g., task manager on Windows, force quit on MacOS), or must fulfill responsibility CR6 above.
CR8	If either CR6 or CR7 are fulfilled, then the infrastructure must also have the ability to cancel the active command with whatever help is available from the active portion of the application (i.e., it must fulfill Responsibilities CR9-CR19 below).
CR9	If the command has invoked any collaborating processes, the collaborating processes must be informed of the cancellation of the invoking command (these processes have their own responsibilities that they must perform in response to this information, possibly treat it as a cancellation.). The information given to collaborating processes may include the request for cancellation, the progress of cancellation, and/or the completion of cancellation.
CR10	If the system is capable of rolling back all changes to the state prior to execution of the command, the system state must be restored to its state prior to execution of the command.
CR11	If the system is not capable of rolling back some of the changes made during the operation of the command prior to cancellation, the system must be restored to a state as close to the state prior to execution of the command as possible.
CR12	If the system is not capable of rolling back some of the changes made during the operation of the command prior to cancellation, the system must inform the user of the difference, if any, between the prior state and the restored state.
CR13	The system must free all resources that it can that were consumed to process the command.
CR14	If some resources has been irrevocably consumed and cannot be restored, the system must inform the user of the partially-restored resources in a manner that they can see it.
CR15	If the command takes longer than 1 second to cancel, control must be returned to the user, if appropriate to the task.
CR16	If control cannot be returned to the user, the system must inform the user of this fact (and ideally, why control cannot be returned).
CR17	The system must estimate the time it will take to cancel within 20%.
CR18	The system must inform the user of this estimate. i. If the estimate is between 1 and 10 seconds, changing the cursor shape is sufficient. ii. If the estimate is more than 10 seconds, and time estimate is within 20%, then a progress indicator is better. iii. If estimate is more than 10 seconds but cannot be estimated accurately, provide other form of feedback to the user.
CR19	Once the cancellation has finished, the system must provide feedback to the user that cancellation is finished (e.g., if cursor was changed to busy indicator, change it back to normal; if progress bar was displayed was displayed, remove it; if dialog box was provided, close it).

Abbildung 6: Liste mit Verantwortlichkeiten für das Szenario Cancel [GJB05]

Für die Anwendung der Taktiken und USAPs ist das implizite und nicht dokumentierte Wissen der Anwender dieser Methode maßgeblich, denn sie leiten nicht genau an, sondern dienen dazu, den Anwender zum Formulieren von Fragen anzuleiten. Damit notwendige Informationen standardisiert erhoben werden können, gibt es für ATAM Vorlagen, Fragelisten und Moderationshinweise.

Wenn Risiken, Nichtrisiken und Tradeoffs beschrieben wurden, werden die Risiken zusammengefasst. Sie resultieren in Risikothemen. Die Risikothemen fassen Risiken inhaltlich zusammen. [CKK02, BCK03]

2.6 BEWERTUNG DER METHODEN SALUTA UND ATAM

BESTIMMUNG VON KONTEXT UND QUALITÄTSMERKMALEN Das betrachtete Qualitätsmerkmal von SALUTA ist Usability, ATAM kann eine Vielzahl von Qualitätsmerkmalen abdecken. Der Kontext der Analyse wird in der Methode ATAM sehr genau betrachtet, allerdings wird der Nutzungskontext hier nicht einbezogen. In SALUTA wird der Nutzungskontext genannt, allerdings wurde in einer Fallstudie ausgesagt, es gäbe

keinen relevanten Nutzungskontext. Da jede Benutzung einer Anwendung in einer bestimmten Situation stattfindet, die als Nutzungskontext bezeichnet wird, sollte dieser hinsichtlich Usability auch immer berücksichtigt werden (siehe Abschnitt über Usability 3.3.3, S. 34).

Bezüglich des Umfangs der Architekturbeschreibung stellt SALUTA keine Anforderungen an die Softwarearchitekturdokumentation. In ATAM wird eine Prüfung gefordert; aber es wird nicht genauer definiert, wann die Softwarearchitektur einen bestimmten Entwicklungsstatus aufweist, mit dem die Analyse durchgeführt werden kann.

BESTIMMUNG UND SELEKTION VON SZENARIOS Die Szenarios von SALUTA beschreiben Interaktionen, die Nutzer mit der Anwendung durchführen können. Sie sind allerdings sehr knapp formuliert, so dass ihr Informationsgehalt minimal ist. Diese abstrakte Darstellung kann zu einem großen Informationsverlust führen, wenn Usability-Anforderungen auf eine Interaktion reduziert werden, ohne andere relevante Qualitätseigenschaften zu beachten. Zudem wird eine Interaktion, die in der Softwarearchitektur zu überprüfen ist, mehrfach genannt und mehrfach bewertet. Dieses Vorgehen verdeckt, wie viele Interaktionen tatsächlich betrachtet werden sollen (Umfang der Evaluation) und führt dazu, dass die architektonische Unterstützung einer Interaktion mehrfach untersucht wird (Aufwand).

Bei der Vorgehensweise von ATAM (Brainstorming von Szenarios) hängt die Operationalisierung von Usability vom implizit vorhandenen Wissen der Anwender dieser Methode ab, d.h. ihrer Fähigkeit, von Usability-Attributen, die für jede Interaktion gelten, konkrete Interaktionen abzuleiten. Da dies nicht einfach ist, wurde ein Workshop aufgesetzt, in dem Stakeholder bestimmte Usability-Konzepte erlernen können [RRD07].

Sowohl SALUTA als auch ATAM verwenden Usability-Attribute auf unübliche Art und Weise. Um den Einfluss eines Szenarios auf Usability zu beschreiben, wird in SALUTA nominal skalierten Usability-Attributen, die für jede Interaktion gelten, eine Rangordnung (Ordinalskala) aufgezwungen. Bedeutung hat dies für die Diskussion im vierten Schritt der Methode, eventuell auch für die Auswahl der Szenarios; denn es wird herausgearbeitet, welche Szenarios analysiert werden sollten. Abgebildet werden aber nur Präferenzen der Nutzergruppen, die leichter verständlich dargestellt werden können. ATAM verwendet Usability-Attribute, um davon Anforderungen abzuleiten, obwohl Usability-Attribute die Qualität jeder Nutzer-System-Interaktion beschreiben und sich nicht auf eine konkrete Anforderung beziehen.

Bei der Auswahl der Szenarios in SALUTA werden keine festgelegten Kriterien zu Rate gezogen. In ATAM ist die Auswahl der Szenarios in der ersten Phase abhängig von der Expertise der teilnehmenden Analysten, in der zweiten Phase vom Abstimmungsverhalten der Teilnehmer. Die Priorisierung und Auswahl von Szenarios ist damit weder in SALUTA noch in ATAM fachlich getrieben oder nachvollziehbar.

Zur Beibehaltung des Fokus leitet die Methode ATAM die Teilnehmer mithilfe eines Rollenmodells dazu an, sich nicht in Diskussionen zu verlieren. Außerdem werden Fragen, die nicht zur Analyse gehören, separat aufgenommen. Bei der Forschungsarbeit zu SALUTA wurden solche Fragen zur praktischen Durchführung der Methode nicht beachtet.

UNTERSTÜTZUNG DER ANALYSE Hintergrundmaterial, Vorlagen und Anleitung der Methode ATAM sind ausführlich. Eine Wissensbasis aus Usability Supporting Architectural Patterns und Taktiken wird bereitgestellt.

SALUTA verfügt über weniger ausgearbeitetes Material und leitet ungenau an, da Details übergangen werden. Insbesondere ist nicht nachvollziehbar, wie die Analyse genau durchgeführt wird, wie sie dokumentiert werden soll und wie die abschließende

Bewertung der Szenarios nachvollziehbar entstehen kann. Statt Vorlagen können einzelne Tabellen der Fallstudien [FvGB04, FB05] in Spezifikationsdokumente übertragen werden.

BESTIMMEN DER ANALYSEERGEBNISSE SALUTA überlässt den Analysten zu große Handlungsspielräume, wenn Szenarios (frei) ausgewählt und (über nicht greifbares Wissen) bewertet werden. In der Methode ATAM werden kreative Techniken eingesetzt, wenn Szenarios erstellt (Brainstorming), ausgewählt (Vorhersage über Komplexität und Abstimmung) und evaluiert werden (Stellen von Fragen). Deshalb hängt die Methode stark von der Expertise der jeweiligen Teilnehmer ab. Somit sind die Ergebnisse beider Methoden schwierig nachvollziehbar und rückvollziehbar.

ZUSAMMENSPIEL VON PROZESSEN In SALUTA kommt es zum Informationsfluss vom Usability Engineering zu den Analysten, allerdings werden hier Usability-Anforderungen stark reduziert, so dass ihr Informationsgehalt minimal wird. Nach Abschluss der Methode kann eine User Evaluation des Endproduktes durchgeführt werden. Auch bei ATAM fließen Informationen vom Usability Engineering zu den Analysten, wenn sie eine Schulung erhalten.

Ein Informationsfluss der Architekten zum Usability Engineering, etablierte Kommunikationsprozesse oder arbeitsteilige Aktivitäten, die definiert und abgestimmt werden, wurden nicht beschrieben.

OFFENE FRAGEN Folgende Fragen bleiben bei SALUTA unbeantwortet:

- Schritt 1: Erstellen von Nutzungsprofilen
  - Wie soll das System auf die Interaktionen reagieren? (Festlegung eines Soll-Verhaltens?)
  - Anhand welcher Kriterien sollen Szenarios ausgewählt werden?
- Schritt 2: Beschreibe die angebotene Usability
  - Wie sollen die erforderlichen Informationen für Szenarios ermittelt werden, zu denen es kein Usability-Pattern bzw. Usability-Property gibt?
  - Wie ist die Softwarearchitektur zu dokumentieren, um den nächsten Schritt vorzubereiten?
- Schritt 3: Evaluiere Szenarios
  - Welche Properties und Patterns sind für ein Szenario wirklich relevant (Soll-Verhalten)?
  - Wie werden die Bewertungen der Szenarios begründet? Wie soll diese Bewertung dokumentiert werden?
- Schritt 4: Interpretiere die Resultate
  - Wie kann bestimmt werden, ob eine Softwarearchitektur Usability im ausreichenden Maße unterstützt?
  - Wie kann bestimmt werden, ob eine Softwarearchitektur Usability nicht in ausreichendem Maße unterstützt?

Hinsichtlich ATAM bleiben folgende Fragen offen:

- Erstellung und Selektion der Szenarios in den Phasen 1 und 2
  - Wie kann abgesichert werden, dass die Auswahl der Szenarios fachlich motiviert ist?
- Analysieren der architektonischen Ansätze anhand von Expertise, Taktiken und Fragekatalogen in den Phasen 1 und 2
  - Welche Fragen sollen Analysten stellen, die nicht über eine sehr gute Kenntnis des Fachgebietes SA, der jeweiligen Qualitätsmerkmale und der Pattern-Literatur verfügen?
- Präsentieren der Ergebnisse in Phase 2
  - Wie können die Ergebnisse mit den Zielen zusammengeführt werden? [KKLN05]

Tabelle 2 auf der nächsten Seite fasst die Bewertung der Methoden zusammen.

Elemente	*SALUTA*	*ATAM*
Bestimmung von Kontext und Qualitätsmerkmalen	(+) Usability im Mittelpunkt (+) Nutzungskontext beachtet	(+) Usability als ein Qualitätsmerkmal unter vielen (+) Umfassende Bestimmung des Kontexts (o) ohne Nutzungskontext, kann aber ergänzt werden
Bestimmung und Selektion von Szenarios	(-) Tripel mit geringem Informationsgehalt (-) Mehrere Szenarios für eine Interaktion (-) Bildung einer Rangfolge von Usability-Attributen, die aber nominal skaliert sind und für jede Interaktion gelten (-) freie Auswahl ohne Anleitung	(+) ausführliche Beschreibung (-) Usability-Attribute als Ausgangspunkt für konkrete Anforderungen (-) Auswahl der Szenarios ist leicht beeinflussbar und wird nicht fachlich begründet
Unterstützung der Analyse	(-) Dokumentation selbstbestimmt (+) Wissensbasis aus Usability-Patterns und Usability-Properties	(+) ausführlich durch Anleitung (+) viele Vorlagen (+) Wissensbasis aus Usability Supporting Architectural Patterns und Taktiken
Bestimmen der Analyseergebnisse	(-) Prüfung, ob jede Usability-Property und jedes Usability-Pattern jedes Szenario unterstützt (-) Bewertung in freier Diskussion mit maximalem Freiheitsgrad (-) nicht gut nachvollziehbar, da Dokumentation selbstbestimmt (-) Rückverfolgbarkeit nicht gewährleistet, da zu viele Freiheiten	(-) Analysten stellen Fragen, die aber unbekannt sind, weil sie von deren Wissen und Erfahrung abhängen (-) wie genau die Analyse geschieht, ist unbekannt (+) gut dokumentiert (-) nicht rückvollziehbar, da Bezug zu einzelnen Szenarios während der Analyse verloren geht
Zusammenspiel von Prozessen	(+) Informationsfluss vom Usability Engineering (-) kein Informationsfluss zum Usability Engineering	(+) Informationsfluss vom Usability Engineering (-) kein Informationsfluss zum Usability Engineering

(+) zweckdienlich, (o) ausreichend, (-) ungenügend hinsichtlich der Ziele dieser Arbeit

Tabelle 2: Inhaltliche Betrachtung der Methoden ATAM und SALUTA

2.7 Forschungsfragen

Basierend auf diesen Erkenntnissen werden folgende Forschungsfragen für die neue Methode SATURN aufgestellt:

Bestimmung von Kontext und Qualitätsmerkmalen

1. *Wie kann der Nutzungskontext in eine szenario-basierte Architekturanalysemethode integriert werden?*

Bestimmung und Selektion von Szenarios

2. *Welches Format eignet sich dafür, Usability-Anforderungen so zu beschreiben, dass sie für die Architekturanalyse ausreichend viele Informationen enthalten und als Kriterien überprüfbar sind?*
3. *Wie können Personen, die keine Usability-Experten sind, dabei unterstützt werden, das Qualitätsmerkmal Usability in Szenarios zu operationalisieren?*
4. *Welche Kriterien eignen sich zur Auswahl der Szenarios? Wie kann methodisch integriert werden, dass die Auswahl fachlich motiviert erfolgt?*

Unterstützung der Analyse

5. *Wie können die Hilfsmittel der Analyse so erstellt werden, dass die Anwender der Methode nicht alle schon existierenden Lösungen (Patterns) kennen oder über ein sehr umfangreiches Fachwissen verfügen müssen?*
6. *Welche Fragetechnik verringert die Freiheitsgrade der Analyse und legt ihr Vorgehen offen?*

Bestimmen der Analyseergebnisse

7. *Wie kann die architektonische Unterstützung von Usability bestimmt und bewertet werden?*

Interdisziplinäre Ausrichtung

8. *Wie kann der iterative und zyklische Prozess zum Entwurf der Architektur methodisch unterstützt werden?*
9. *Wie kann die interdisziplinäre Kooperation zwischen den beteiligten Prozessen unterstützt werden?*

2.8 Zusammenfassung

Dieses Kapitel untersuchte ausführlich, inwiefern frühere Arbeiten den gestellten Anforderungen an diese Forschungsarbeit genügen. Die jeweiligen Arbeitsschritte von SALUTA und ATAM, die zugrunde liegenden Theorien, die Argumentationen, die eigentliche Analyse der Architektur mit einem konkreten Pattern-Set bzw. durch unbekannte Fragen, die aus Erfahrungen oder von bekannten Patterns abgeleitet wurden, führten zu einer inhaltlichen Bewertung.

Aus offenen Fragen zu den beiden früheren Methoden wurden neun Forschungsfragen für diese Arbeit abgeleitet. Diese beziehen sich zum einen auf die neu zu erstellende Methode und zum anderen auf neu zu erstellende Szenarios.

Die methodischen Fragen über (1) die Integration des Nutzungskontextes, (3) die Operationalisierung des Qualitätsmerkmales Usability durch Personen, die keine

Usability-Experten sind, (4) Kriterien zur Auswahl von Szenarios, (6) die verwendete Fragetechnik, (7) die Bewertung der architektonischen Unterstützung für Usability sowie die (8) iterative und (9) interdisziplinäre Ausrichtung der Methode werden in Kapitel 4 (ab S. 48) behandelt. Das folgende Kapitel 3 beantwortet die Frage über (2) ein geeignetes Format für Szenarios und bereitet Antworten auf die Fragen (3) und (4) vor, indem die entsprechenden verwandten Arbeiten auch unter diesem Aspekt untersucht werden. Es stellt (5) das Hilfsmittel vor, mit dem die neue Softwarearchitekturanalysemethode von einem umfangreichen Wissen über Patterns und Best Practices unabhängig wird.

3 INTERAKTIONSSZENARIOS FÜR DIE METHODE SATURN

Zuerst wird erklärt, warum Interaktionsszenarios erarbeitet werden (Abschnitt 3.1). Daraufhin stellt Abschnitt 3.2 verwandte Arbeiten zu Szenarios und zur Erhebung von Szenarios vor. Wie die Interaktionsszenarios für die Methode SATURN definiert und klassifiziert werden, erläutert Abschnitt 3.3. Abschnitt 3.4 behandelt, wie die Interaktionsszenarios erstellt werden, Abschnitt 3.5 zeigt, wie sie verwaltet werden. Zum Abschluss fasst Abschnitt 3.6 das Kapitel zusammen. Der Anhang enthält schließlich fünfzig Interaktionsszenarios (Abschnitt A.2, S. 204 ff.).

3.1 MOTIVATION

Dieses Kapitel thematisiert die Fragen zwei, drei, fünf und neun der Forschungsfragen (Abschnitt 2.6, S. 23ff.).

Forschungsfrage 2 betrifft das Format, mit dem Usability-Anforderungen so beschrieben werden, dass sie in einer Softwarearchitekturanalysemethode verwendet werden können. In szenario-basierten Softwarearchitekturanalysemethoden werden die Anforderungen eines Qualitätsmerkmals mit Szenarios operationalisiert [DN02, BG04]. Szenarios beschreiben ein Modell möglicher Ereignisse [WKRSS09], also die hypothetische Durchführung einer Sache.[1] Sie werden als Testfälle verwendet, um zu prüfen, ob sie mit einer bestimmten Softwarearchitektur durchgeführt werden können. Für die Methode SATURN wird ein Format für Szenarios benötigt, das ausreichend viele Informationen für die Architekturanalyse bereithalten soll, die als Kriterien überprüfbar sind.

Außerdem sollen (Forschungsfrage 3) die Anwender dieser Methode (meist Architekten) Szenarios für die Architekturanalyse ohne detailliertes Vorwissen zu Usability ermitteln können.

Es soll auch möglich sein, diese Architekturanalyse ohne detaillierte Kenntnisse von Softwarearchitektur-Patterns durchzuführen (Forschungsfrage 5).

Nicht zuletzt soll mit den Szenarios auch die Kooperation zwischen Usability-Engineering und Softwarearchitektur unterstützt werden (Forschungsfrage 9).

Das Kapitel stellt jene Interaktionsszenarios vor, welche Interaktionen beschreiben, die Architekturänderungen erfordern können, sofern sie noch nicht in der Softwarearchitektur berücksichtigt wurden. Um den Aufwand, der zur Erstellung von Interaktionsszenarios benötigt wird, zu verringern, wird außerdem ein Katalog von Interaktionsszenarios für die Methode SATURN erarbeitet und bereitgestellt.

3.2 VERWANDTE ARBEITEN

3.2.1 *Szenarios*

„General Usability Scenarios“ (GUS) von Bass et al. [BJK01] beschreiben Interaktionen, die in Anwendungssoftware zur Laufzeit auftreten können und wie das System darauf reagieren soll: sie skizzieren konkrete Anwendungsbeispiele und beschreiben einen Architekturmechanismus, der dazu beiträgt, die Szenarios aus Usability- und SA-Perspektive umzusetzen und zu kategorisieren. Dazu nutzen sie ein hierarchisches Usability-Modell. Diese „Usability Benefit Hierarchy“ ist als mehrstufige Baumstruktur von Usability-Attributen dargestellt und beschreibt die Usability-Unterstützung

1 Duden, 18.4.2012, http://www.duden.de/rechtschreibung/Szenario

durch den Einsatz der Szenarios. Der Nutzen „increases individual user effectiveness“ entspricht dabei beispielsweise dem Usability-Attribut „Effektivität“ von Shackel [SR91] oder Preece et al. [PRS07]. Außerdem beschreibt eine Hierarchie von Softwarearchitektur-Mechanismen, wie relevant ein Szenario für die SA ist. Die Interaktionsszenarios von SATURN beschreiben konkrete Usability-Anforderungen mit Hilfe von Interaktionen als Stimulus und der Response des Systems. Sie sind entsprechend ihrer Verwendung in dieser Methode auf verschiedene Art und Weise klassifiziert. Sie bedienen zudem ein breiteres Spektrum an möglichen Interaktionen als [BJK01], denn es wurden weitere Quellen hinzugezogen und damit neue Interaktionen zur Diskussion und zur Analyse bereitgestellt: es gibt 50 generische Interaktionsszenarios mit und ohne offensichtlichen Einfluss auf die Architektur. Langfristig können Analysten und Architekten somit Usability-Anforderungen mit Architekturbezug identifizieren und prüfen.

Rafla et al. [RRD07] zeigen, dass die kurzen Beispiele in den Szenarios von [BJK01] nicht ausreichend selbsterklärend sind. Deshalb verweisen die hier erstellten Szenarios auf die Patterns, von denen sie abgeleitet wurden und erläutern sie kurz. Mit diesen Informationen kann die Beschreibung von konkreten Anforderungen unterstützt werden.

Golden et al. [GJB05] erweiterten die Szenarios von [BJK01] durch eine Liste von Verantwortlichkeiten und ein Fallbeispiel zum Einsatz in der szenario-basierten Softwarearchitekturanalyse ATAM. Diese Ergänzungen werden bei den hier erstellten Szenarios referenziert, sind also auch während SATURN schnell verfügbar und tragen zum besseren Verständnis eines Interaktionsszenarios bei.

Juristo et al. [JMSS07] stellen „Usability-Elicitation-Patterns“ vor, um funktionale Usability-Anforderungen für die SA mit Usability-Nichtexperten zu erheben. Dazu wurden zuerst Usability-Patterns verschiedener Autoren (beispielsweise Tidwell [Tid05], van Welie [vW12]) hinsichtlich beschriebener Usability-Mechanismen analysiert. Für jeden Mechanismus wurde ein Elicitation-Pattern beschrieben, das Fragen zur Spezifikation detaillierter Anforderungen für das System auflistet. Probleme, Kontext und Erklärungen zu den Fragen basieren auf Usability-Patterns. Die Usability-Elicitation-Patterns dienen als eine Checkliste während der Erhebung von Usability-Anforderungen. Allerdings müssen auch hier Usability-Konzepte speziell erlernt und angewandt werden.

Schmettow und Niebuhr [SN07] setzen Patterns zur Analyse ein. Sie stellen eine Pattern-basierte Usability-Inspektions-Methode für Softwareentwickler vor. Patterns werden über 16 Kategorien von benutzerinitiierten Interaktionen klassifiziert. Die gute Anwendbarkeit der Inspektionsmethode wird damit begründet, das die Aufmerksamkeit von Usability-Problemen zu konstruktiven Lösungen gelenkt ist. Auch in SATURN werden Szenarios basierend auf der Benutzer-Perspektive kategorisiert. Durch die Analyse wird ein konstruktiver Beitrag zum Design von Interaktionen geleistet. Der in dieser Arbeit verfolgte Ansatz geht etwas weiter: Weitere Vereinfachungen erreichen die SATURN-Szenarios, da sie durch die Sprache und mit dem Standpunkt der Benutzer Interaktionen beschreiben, anstatt Patterns anzubieten, deren Inhalte erst verstanden werden müssen. Es werden zudem sowohl benutzer- als auch systeminitiierte Interaktionen einbezogen. Außerdem eignen sich die Interaktionsszenarios als Vorlage für konkrete Anforderungen und werden auch so verwendet.

Die „Problem Frames“ von Jackson [Jac01] verwenden „frame diagrams“, um Probleme darzustellen und zu strukturieren. Dabei werden bestehende Problem Frames instanziiert und Probleme konstruktiv beschrieben. Problem Frames eignen sich, um Probleme zu beschreiben, die in Architekturpatterns enthalten sind [CHH05]. Specker und Wentzlaff [SW07] schlagen „HCI-Frames“ vor, die Problemkategorien durch Usability-Funktionalitäten ergänzen. Für den Einsatz in SATURN sind die Problem Frames nicht gut geeignet, da sie eine formale Sprache darstellen. Sie führen zwar zu einer präzisen

Problembeschreibung und erlauben damit ein akkurates Verständnis von bestimmten Teilen des Problems, aber die Teilnehmer einer Softwarearchitekturanalyse müssten eine formale Sprache erlernen und sich neues fachliches Wissen und entsprechend abstraktes Denken aneignen. Wenn sich Stakeholder nicht regelmäßig mit der Softwareentwicklung befassen, werden ihnen die Kenntnis und das Verständnis der formalen Sprache nicht weiter nützlich sein. Dieser Ansatz stellt für die Zielgruppen Softwaretechniker und Stakeholder keinen intuitiven und sofort verständlichen Ansatz dar.

Essential Use Cases [CL99] betrachten den Zweck einer Interaktion, die Verantwortlichkeit des Systems einerseits und die schrittweise Nutzer-System-Interaktion andererseits. Um Essential Use Cases auf einem guten Abstraktionsgrad zu beschreiben, ist Expertenwissen im Bereich Usability notwendig; zudem sind sie sehr kleinteilig (siehe Beispiel), weshalb auch Essential Use Cases nicht für die SAA ohne Usability-Experten geeignet sind.

Essential Use Case		*Detailed Use Case*	
Bargeld auszahlen		*Bargeld auszahlen*	
Zweck	*Verantwortlichkeit des Systems*	*Aktion*	*Response*
sich identifizieren		Karte einstecken	
	Identität verifizieren, Auswahl anbieten		Magnetstreifen lesen PIN abfragen
Auswahl treffen		PIN eingeben	
	Bargeld ausgeben		PIN verifizieren Anzeigen des Menüs „Transaktionsoptionen“
Bargeld nehmen		Taste drücken	
			Betrag abfragen
		Betrag eingeben	
			Betrag anzeigen
		Taste drücken	
			Karte geben
		Karte nehmen	
			Bargeld ausgeben
		Bargeld nehmen	

Tabelle 3: Essential Use Cases und dazu gehörender Detailed Use Case ([CL99] zitiert in [FJ06])

Use Cases [Obj11] werden in der Objektorientierten Analyse sowie im Objektorientierten Design verwendet und sind daher nah an der Implementierung [FJ06]. Sie betrachten einen Nutzfall, eine Aktion, die mit der Anwendung durchgeführt werden kann. Das System wird dabei als Black Box betrachtet. Bei den Interaktionsszenarios soll aber die Response des Systems beschrieben werden. Daher ist das Format der Use Cases ungeeignet dafür, die abstrakten Interaktionsszenarios zu beschreiben; sie sind aber aufgrund ihrer typischen Anwendungsdomäne dafür geeignet, bei der Evaluation von Interaktionsszenarios den Fokus auf die jeweilige technische Umsetzung zu lenken.

3.2.2 *Erhebung von Szenarios*

Der „Quality-Attribute-Workshop“ von Barbacci et al. [BEL+03] ist eine Schulung für Stakeholder, um zu erlernen, wie Szenarios von Qualitätsattributen abgeleitet werden können. Nachdem wesentliche Qualitätsattribute einer Software identifiziert wurden, führt ein Brainstorming zu Qualitätsanforderungen, die für eine Softwarearchitekturanalyse als Szenarios formuliert werden. Allerdings kann fehlendes Fachwissen zu den einzelnen Qualitätsattributen die Qualität der Ergebnisse beeinflussen.

Rafla et al. [RRD07] stellen deshalb einen „Usability-Quality-Attribute-Workshop“ vor, der darauf spezialisiert ist, Usability-Anforderungen auszuwählen. In einem zweistufigen Vorgehensmodell werden die Stakeholder unterrichtet und erlernen das Konzept der „Usability-Property“ von Folmer et al. [FGB03] sowie ein erläuterndes Beispiel. Für jede Usability-Property wird ein Szenario für die zu entwerfende Softwarearchitektur erstellt. In der zweiten Stufe erlernen die Stakeholder die „General Usability Scenarios“, ebenfalls unterstützt durch Beispiele. Für jedes Szenario wird entschieden, ob es für die Software zutrifft. Wenn ja, wird ein konkretes Szenario formuliert. Während dieser Arbeitsschritte sichern Usability-Experten ab, dass die grundlegenden Konzepte verstanden wurden, dass die neuen Szenarios valide sind und dass sie sich für die zu entwickelnde Software eignen. Nach einer Diskussion, in der Szenarios konsolidiert und priorisiert werden, beschreiben die Analysten die Szenarios detaillierter und ergänzen Anwendungsfälle. Aus verschiedenen Gründen wäre dieses Vorgehen für die Methode SATURN problematisch:

- Stakeholder können sich nicht auf ihre Probleme konzentrieren, da sie Usability-Konzepte erlernen und trainieren müssen.
- Von Beginn an wird die Suche nach Anforderungen auf bestimmte Forschungsergebnisse begrenzt.
- Der Hauptbeitrag wird nicht durch die Teilnehmer geleistet, sondern durch Usability-Analysten, die zwar eigenes Erfahrungswissen haben, aber wegen ihrer geschulten Wahrnehmung nicht mehr unvoreingenommen auf das Problem blicken können. Außerdem wird so eine notwendige Lernerfahrung zum Einsatz des erworbenen Wissens, beispielsweise für spätere Analysen ohne die Analysten, nicht erleichtert.

Deshalb wird in SATURN kein Workshop durchgeführt, so dass die Teilnehmer ihr Wissen über die Problemdomäne so weit wie möglich unvoreingenommenen mitteilen können.

3.3.1 *Definition eines Interaktionsszenarios*

Interaktionen umfassen jede zielgerichtete Art von Kommunikation zwischen Mensch und Maschine [DFAB03]. Die Interaktionsszenarios von SATURN (Abbildung 7) beschreiben Nutzer-System-Interaktionen, die Architekturänderungen erfordern können, wenn sie noch nicht in der Softwarearchitektur berücksichtigt wurden. Sie beschreiben eine von System oder Nutzer gestartete Interaktion und die daraufhin erwartete Reaktion: Eine *Quelle* (Benutzer oder System) initiiert eine Interaktion und sendet damit einen *Stimulus* an eine Anwendung (*Artefakt*) in einer bestimmten *Umgebung* und erwartet eine bestimmte Reaktion darauf (*Response*), deren erfolgreiche Realisierung bestimmten Anforderungen genügen muss, die mit bestimmten Prüfungen kontrolliert werden können (*Response-Prüfung*).

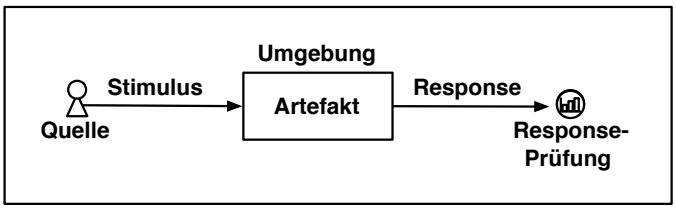

Abbildung 7: Interaktionsszenario

Die Kernelemente eines Interaktionsszenarios werden in Tabelle 4 aufgelistet und definiert. Ihre Definition orientiert sich an dem Format der Szenarios von Bass et al. [BCK03], wobei der Hauptunterschied darin liegt, dass der Stimulus der Szenarios von [BCK03] Wünsche und Erwartungen von Nutzern beschreibt (effiziente Nutzung), während die Interaktionsszenarios von SATURN konkrete Interaktionen nennen. Stimulus, Umgebung und Response-Prüfung wurden anders definiert, eine Klassifikation wurde ergänzt.

Name	*Beschreibung*
Nummer*	Laufende Nummer des Interaktionsszenarios
Kurzname*	Charakteristischer Name als Kurzreferenz, der meist die Response umreißt; Name kann geändert werden
Quelle	Eine den Stimulus generierende Einheit, beispielsweise Benutzer (Kontextfaktor Benutzertypen), Softwaresystem oder ein Teil eines Softwaresystems (Kontextfaktor Mobile Anwendung)
Stimulus*	Initiierte Nutzer-System-Interaktion (Kontextfaktor Aufgabe)
Umgebung*	Konditionen, unter denen ein Stimulus auftritt (beispielsweise Laufzeit, Überlast), bestimmte Umgebungstypen; Umgebungen, in denen die Anwendung benutzt wird; ebenso Laufzeitumgebung, d. h. Systemstatus wie z. B. ungewöhnliche Konditionen wie hohe Last, Ausfall eines Prozessors (Kontextfaktoren Umgebung, Endgerät, Mobile Anwendung, siehe Abschnitt 4.1.2, S. 48ff.)
Artefakt	Das stimulierte Artefakt: das ganze System oder nur ein (oder mehrere) Teil(e) davon (Hardware und/oder Software, Kontextfaktor Endgerät, Mobile Anwendung)
Response	Die Aktivität des Artefakts nach Eintreffen des Stimulus
Response-Prüfung*	Messung oder andere Prüfung der Umsetzung der Anforderung; Analyse der Response, entweder durch Inspektion durch Experten (A für Analyse) oder Evaluation mit Benutzern (U für User-Evaluation)
Usability-Pattern(s)*	Referenz zu dem/den Pattern(s), aus denen das Interaktionsszenario extrahiert wurde; mit einer kurzen Erläuterung zur Anforderung
* neue bzw. neu definierte Elemente im Vergleich zu den Szenarios von Bass et al. [BCK03]	

Tabelle 4: Kernelemente eines Interaktionsszenarios

Zentrale Elemente eines Interaktionsszenarios sind Stimulus und Response; ein Stimulus kann mehrere Responses haben; es können also verschiedene Interaktionsszenarios mit dem gleichen Stimulus vorkommen.

Die Angaben zur Response-Prüfung verweisen auf verschiedene Evaluationsmethoden, da es viele Möglichkeiten zur Evaluation der Response eines Artefakts auf den Stimulus gibt[2]. Nach ihrer Einbeziehung von Benutzern kategorisiert, lassen sich Inspektionen durch Experten (A für Analyse) und Evaluationen durch Endbenutzer (U für User-Evaluation) unterscheiden. Bei User-Evaluationen werden Usability-Probleme durch Anwendungsfehler und menschliche Fehler aufgedeckt. Bei Inspektionen durch Experten wird das Design des User Interfaces, der Interaktionen oder der Softwarearchitektur analysiert und evaluiert, um Ursachen und Indikatoren für mögliche Usability-Probleme offenzulegen.

Die Interaktionsszenarios werden von einem oder mehreren Patterns abgeleitet (Abschnitt 3.4.1, S. 39). Ein Pattern wird referenziert, damit seine Inhalte während der Evaluation des betreffenden Interaktionsszenarios im Bedarfsfall nachgelesen werden können. Damit die Abstraktionsebene der Interaktionsszenarios erhalten bleibt, werden nicht Lösungen, sondern die im Pattern beschriebene Anforderung in ein bis zwei Sätzen kurz zusammengefasst.

2 Einen Überblick über Usability-Evaluationsmethoden aus softwaretechnischer Perspektive bieten [FJM05].

Fünfzig Interaktionsszenarios werden im Anhang aufgeführt (Abschnitt A.2, S. 204ff.). Während einer Softwarearchitekturanalysemethode wird ein Set von Interaktionsszenarios ausgewählt, das Interaktionen mit der analysierten Anwendung beschreibt. Die Interaktionsszenarios enthalten Voreinträge für fast alle Elemente, nur Umgebung und Artefakt wurden ausgelassen, da diese ausschließlich durch den konkreten Nutzungskontext einer mobilen Anwendung bestimmt werden.

Damit die Interaktionsszenarios gut aufgefunden werden können, um aufzuzeigen, welchen Einfluss sie auf die Usability und möglicherweise auf die Softwarearchitektur haben, werden sie entsprechend klassifiziert. Die Tabelle 5 nennt die Klassifikationen, die in den folgenden Abschnitten erläutert werden.

Name	*Beschreibung*
Interaktions-kategorie(n)	Klassen von Interaktionsszenarios
Usability-Ziel(e)	Begründete Beeinflussung von Usability-Zielen durch das Interaktionsszenario (Effektivität, Effizienz, Sicherheit, Nützlichkeit, Erlernbarkeit, Einprägsamkeit)
Potenzielle Architektur-sensitivität	Anhand der Patterns und/oder Expertenmeinung bestimmter potenzieller Änderungsaufwand in der Architektur, falls das Interaktionsszenario nachimplementiert werden muss.

Tabelle 5: Elemente, die ein Interaktionsszenario klassifizieren

3.3.2 *Klassifikation nach Interaktionskategorien*

Damit nicht immer der komplette Katalog von Interaktionsszenarios während der Softwarearchitekturanalysemethode durchsucht werden muss und Interaktionsszenarios dennoch leicht auffindbar sind, wurden sie nach ihren Inhalten in folgende 13 Interaktionskategorien gruppiert. Sie sind aus Nutzersicht verfasst. [Sei09]

I_1. Erfassung und Behandlung von Benutzereingaben

I_2. Orientierung

I_3. Navigieren, Browsen, Auswählen

I_4. Ausführen, Wiederholen und Zurücknehmen von Befehlen

I_5. Interagieren mit unbekannten Systemen

I_6. Selektion und Exploration von Daten

I_7. Bearbeiten von Grafiken und grafischen Objekten

I_8. Austausch und Manipulation von Daten

I_9. Informationsbeschaffung

I_{10}. Strukturieren und Anzeigen von Content, Informationen, Daten

I_{11}. Behandlung von Fehlern und Hilfe

I_{12}. Anpassen durch Benutzer, für Benutzer, für Aufgaben

I_{13}. Kooperation (in einem System, mit anderen Systemen)

Interaktionsszenarios können unter verschiedenen Aspekten betrachtet werden, weshalb sie sich auch mehreren Interaktionskategorien zuordnen lassen. Beispielsweise befasst sich das Interaktionsszenario *Prüfen auf Korrektheit (Checking for Correctness)* (Tabelle 75, S. 218) mit der Aufbereitung von Benutzerdaten und mit Mechanismen zur Fehlertoleranz, weshalb es in die Interaktionskategorien *Erfassung und Behandlung von Benutzereingaben* und *Fehlerbehandlung und Hilfe* eingruppiert wird. Je nach Blickwinkel einer Person (Benutzereingaben, Fehlerbehandlung oder beides) kann sie sich das relevante Interaktionsszenario herausfiltern.

Tabelle 6 zeigt, wie die aktuellen Interaktionsszenarios über Interaktionskategorien verteilt sind.

Interaktionskategorie	*Anzahl*
1. Erfassung und Behandlung von Benutzereingaben	13
2. Orientierung	3
3. Navigieren, Browsen, Auswählen	3
4. Ausführen, Wiederholen und Zurücknehmen von Befehlen	9
5. Interagieren mit unbekannten Systemen	3
6. Selektion und Exploration von Daten	3
7. Bearbeiten von Grafiken und grafischen Objekten	0
8. Austausch und Manipulation von Daten	5
9. Informationsbeschaffung	2
10. Strukturieren und Anzeigen von Content, Informationen, Daten	11
11. Behandlung von Fehlern und Hilfe	9
12. Anpassen durch Benutzer, für Benutzer, für Aufgaben	12
13. Kooperation (in einem System, mit anderen Systemen)	7

Tabelle 6: Verteilung der Interaktionsszenarios über Interaktionskategorien

3.3.3 *Klassifikation nach Usability-Ziel*

Um zu begründen, warum ein Interaktionsszenario sinnvoll ist, d. h. wie es Usability beeinflussen kann, werden Usability-Attribute als Ziele aufgeführt.

Preece et al. [PRS07] verstehen Usability als Teilziel der User Experience (Benutzungserlebnis). Dieser Begriff ist in der (untersuchten) wissenschaftlichen Literatur nicht konkret definiert. User Experience beschreibt, wie die Benutzer die Interaktion mit einer Anwendung subjektiv erleben: ob die Benutzung ihrer Meinung nach ästhetisch ist, inwiefern sie Spaß und Freude empfinden, inwiefern sie zufrieden sind. Zufriedenheit zählt hier also zur User Experience, nicht zur Usability, da Norman [Nor04] nachwies, dass Ästhetik und Zufriedenheit zusammenhängen, denn er fand eine positive Korrelation zwischen diesen beiden Komponenten. Auch die Usability einer Anwendung beeinflusst die Zufriedenheit der Benutzer: wenn eine Anwendung schwierig zu erlernen ist, mehrfach die gleichen Eingaben abfragt und die Daten der Benutzer nicht sicher sind, sind die Benutzer mit dieser Anwendung unzufrieden. Das heißt, wenn ein Usability-Ziel nicht unterstützt wird, leidet auch die User Experience insgesamt. Da Zufriedenheit der Nutzer nur mit Nutzern gemessen werden kann und bei der Softwarearchitekturanalysemethode kein User Interface, sondern die Architektur untersucht wird, ist es auch aus diesem Grund (im Gegensatz zu Definitionen von [Deu11, Nie93, SR91]) sinnvoll, hier das Attribut Zufriedenheit auszuschließen und der Argumentation von [PRS07] zu folgen.

Tabelle 7 stellt die typischen Usability-Definitionen gegenüber, um noch einmal zu verdeutlichen, dass die verwendete Usability-Definition von Preece et al. [PRS07] die Leistung akzentuiert und die Meinungsäußerung (z. B. zur Zufriedenheit) „auslagert“ und dabei auch in einen größeren Zusammenhang stellt.

ISO 9241-11	*Shackel*	*Nielsen*	*Preece et al.*
Leistung der Benutzer			
Effektivität Effizienz	Effektivität Erlernbarkeit Flexibilität	Erlernbarkeit Effizienz Einprägsamkeit Fehlertoleranz	Effektivität Effizienz Sicherheit bei der Benutzung Nützlichkeit Erlernbarkeit Einprägsamkeit
Meinung der Benutzer			
Zufriedenheit	Attitüde	Zufriedenheit	*Abgrenzung: Usability ist Teilziel der User Experience, zu dieser zählen u.a. auch Zufriedenheit und Ästhetik*

Tabelle 7: Usability-Definitionen im Vergleich

Folgende Usability-Attribute werden also für die Interaktionsszenarios definiert [PRS07]:

- Effizienz: welche Ressourcen Nutzer aufwenden müssen, um ihre Aufgaben zu erledigen; wie gut das System sie dabei unterstützt,
- Effektivität: wie gut Nutzer mithilfe des Systems ihre Aufgaben erledigen können; wie zuverlässig das System arbeitet,
- Sicherheit bei der Benutzung: wie ein Produkt seine Benutzer und ihre Daten vor gefährlichen oder anderen unerwünschten Situationen schützt,
- Nützlichkeit: wie gut das System den richtigen Nutzen für eine bestimmte Benutzergruppe bereithält,
- Erlernbarkeit: wie einfach ein System von Nutzern erlernt werden kann,
- Einprägsamkeit: wie gut sich Nutzer daran erinnern können, wie ein Produkt zu bedienen ist, nachdem sie es einmal gelernt haben.

In den Interaktionsszenarios wird jeweils kurz erläutert, warum ein Interaktionsszenario ein bestimmtes Ziel unterstützt (dies ist meist der Fall und wird mit einem vorangestellten (+) ausgedrückt) oder nicht (vorangestellt wird dann das Symbol (-)).

Beispielsweise nennt das schon erwähnte Interaktionsszenario Nr. 14 *Prüfen auf Korrektheit (Checking for Correctness)* (Tabelle 75, S. 218) folgende Beeinflussung von Usability-Zielen:

(+) Effektivität: Das System unterstützt die Benutzer bei der korrekten Durchführung ihrer Aufgaben.

(+) Sicherheit: Die gespeicherten Daten der Nutzer sind in dem Maße korrekt und valide, wie das System die Prüfung durchführt. Es werden bestimmte Fehler vermieden.

Tabelle 8 stellt die Häufigkeitsverteilung der Usability-Ziele dar: 36 Interaktionsszenarios unterstützen Effizienz, acht Effektivität, 18 Sicherheit, 13 Nützlichkeit, zwölf Erlernbarkeit, zwei Einprägsamkeit.

Usability-Ziel	*Anzahl*
Effizienz	36
Effektivität	8
Sicherheit	18
Nützlichkeit	13
Erlernbarkeit	12
Einprägsamkeit	2

Tabelle 8: Verteilung der Usability-Attribute in Interaktionsszenarios

3.3.4 *Klassifikation nach möglichen Architekturänderungen*

Die Aufwandsschätzung von Nachimplementierungen in einer Architektur basiert zum einen auf den Patterns, von denen ein Interaktionsszenario abgeleitet wurde (siehe Abschnitt 3.4.1), und zum anderen auf der Einschätzung von Experten. Tabelle 9 zeigt die drei Typen von Aufwandsschätzungen von Nachimplementierungen in einer Architektur auf, wie sie im Rahmen der Methode SATURN definiert sind.

Die Benotung erfolgt je nach Änderungsaufwand: keine Änderungen werden als Typ 0 bezeichnet, interne Änderungen von architektonischen Elementen, die zu Verhaltensänderungen führen, aber nicht die Struktur ändern, werden als Typ 1 bezeichnet, Typ 2 umfasst komplexe Modifikationen durch das Ergänzen oder Entfernen von architektonischen Elementen. Die potenzielle Architektursensitivität eines Interaktionsszenarios basiert initial auf den Lösungen in den Patterns und den Aussagen in der Literatur und langfristig auf der Bewertung der Interaktionsszenarios in den Architekturanalysen.

Der tatsächliche Aufwand für die nachträgliche Implementierung der jeweiligen Anforderung kann nur anhand einer konkreten Softwarearchitektur, ihrer aktuellen Unterstützung für die Anforderung und der jeweiligen Architekturebene beurteilt werden. Je nach vorhandenen Elementen, ihrer Struktur und ihren Kollaborationen, müssen Elemente und Kollaborationen ergänzt, entfernt oder modifiziert werden. Deshalb ist diese Kennzahl zur Aufwandsschätzung von Nachimplementierungen in einer Architektur nur ein Indiz für den möglichen Modifikationsaufwand aus Sicht des Architekten, der die Anwendung analysiert. Die Klassifikation des Interaktionsszenarios wird deshalb nach jeder Analyse überprüft und gegebenenfalls angepasst.

Typ	*Beschreibung*	*Änderungsaufwand*
0	Keine Modifikationen	keine Änderungen notwendig
1	Einfache Modifikationen	Verhalten einer oder mehrerer Komponenten und/oder Schnittstellen soll geändert werden; Modifikationen erfordern keine strukturellen Änderungen
2	Komplexe Modifikationen	Ein oder mehrere architektonische Elemente müssen hinzugefügt, entfernt oder anders kombiniert werden; Modifikationen betreffen sowohl Struktur als auch Verhalten der architektonischen Elemente
3	Unvorhersehbare Komplexität	Unbekannter Aufwand, kann von Typ 1 oder 2 sein, nicht jedoch von Typ 0

Tabelle 9: Komplexität von architektonischen Änderungen

Um anzugeben, wie sicher diese Einschätzung ist, werden zwei Ansätze kombiniert: es wird die Häufigkeit von architektonischen Änderungen für ein Interaktionsszenario gezählt und eine Expertenmeinung eingeholt. Diese Einschätzung des Vertrauens in ein Interaktionsszenario wird als Vertrauensfaktor („trust factor“) bezeichnet und wurde für Patterns definiert [BGG08].

Basiert die Einschätzung der potenziellen Architektursensitivität auf wissenschaftlichen Quellen, erhält ein Interaktionsszenario drei Sternchen „***“, ist diese Einschätzung durch mehrere Einsätze des Interaktionsszenarios gesichert, erhält es zwei Sternchen „**“, ein durch einen Experten bewertetes Interaktionsszenario erhält ein Sternchen „*“, neue Interaktionsszenarios bleiben ohne Sternchen (siehe Tabelle 10). Dieses Vorgehen stimmt mit dem von Alexander [AIS78] vorgeschlagenen Signifikanzwert überein, der von Null bis zwei Sternchen angab, wie beständig ein von ihm beschriebenes Pattern ist.

Vertrauensfaktor	*Beschreibung*
ohne Zeichen	Noch von einem Experten zu prüfendes Interaktionsszenario.
*	Neue Einschätzung durch einen Experten.
**	Mehrere weitere Beispiele zeigen, dass die Einschätzung grundsätzlich zutrifft.
***	Mehrere Beispiele und Dokumente (z.B. Veröffentlichungen) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 10: Vertrauensfaktor für Bewertungen

Zum Beispiel gibt es beim Interaktionsszenario Nr. 12 *Auto-Komplettierung* (Tabelle 73, S. 216) mehrere Implementierungsmöglichkeiten. Es könnte eine neue und zu ergänzende Komponente erstellt werden. Es ist aber auch möglich, eine bestehende Komponente zu erweitern und die neue Funktionalität mit anderen Komponenten zu integrieren. In beiden Fällen hängt der erforderliche Änderungsaufwand der Architektur von der Anzahl zu erweiternder Eingabefelder, der Anzahl an Quellen mit Vokabeln für die Vorschläge zur Vervollständigung und von der Anzahl an Quellen mit Kontextinformationen (falls Eingabe-vorschläge kontextabhängig sind) ab. Es ist natürlich auch möglich, die Funktionalität so zu implementieren, dass einzelne Eingabefelder mit kontextunabhängiger Vervollständigung erweitert werden, die als einzige Quelle

Vorschlagsdaten enthält. Die Voraussetzungen für Typ 1 und 2 sind daher grundsätzlich erfüllt, es erfolgt die Klassifikation als Typ 3. Da diese Einschätzung die Meinung eines Experten wiedergibt, der von allen drei Implementierungen mehrere Beispiele kennt, kann eine Klassifikation von 3** angegeben werden.

Ein weiteres Beispiel ist das Interaktionsszenario Nr. 4 *Abbrechen* (Tabelle 65, S. 208). Anhand der Quellen [BJK01, GJB05] ist aufgrund der notwendigen Ergänzung von mehreren Elementen und neuen Kollaborationen die potenzielle Architektursensitivität von 2 wissenschaftlich belegt. Die Einschätzung der Architektursensitivität des Interaktionsszenarios kann aufgrund seiner Quellenlage den höchsten Vertrauensfaktor erhalten: die Gesamtbewertung ist also 2***.

Im Allgemeinen ist die Architektursensitivität der „Architecture-sensitive Usability Patterns“ [BJK01] und der „Bridging Patterns“ von Folmer et al. [FvWB06] nachgewiesen. Deshalb erhalten alle davon abgeleiteten Interaktionsszenarios die potenzielle Architektursensitivität 2***. Die anderen Interaktionsszenarios erhalten je nach Quellenlage eine Bewertung durch einen Experten.

Tabelle 11 fasst zusammen, welche Architektursensitivität die Interaktionsszenarios haben können, sofern sie noch nicht in einer Softwarearchitektur berücksichtigt wurden. So weisen zwölf Interaktionsszenarios auf einfache Modifikationen hin, 37 auf komplexe Modifikationen; ein Interaktionsszenario (Auto-Komplettierung) kann zu diesem Zeitpunkt der einen oder anderen Kategorie zugeordnet werden.

Typ	*Beschreibung*	*Anzahl von Interaktions-szenarios*
1	Einfache Modifikationen	12
2	Komplexe Modifikationen	37
3	Unvorhersehbare Komplexität	1

Tabelle 11: Potenzielle Architektursensitivität der Interaktionsszenarios

Die folgenden Abschnitte erläutern, wie valide Interaktionsszenarios erstellt und für den langfristigen Aufbau einer Wissensbasis gepflegt werden können.

3.4.1 *Extraktionsmethode und Beispiel*

Zhu et al. [ZBJ04] extrahierten systematisch Textabschnitte aus Architekturpatterns (siehe nächster Abschnitt zu Patterns) und erstellten Szenarios für die szenario-basierte Softwarearchitekturanalysemethode ATAM. Hier wird ihr Ansatz auf Interaktionspatterns angewendet, um Interaktionsszenarios nach der Definition in Abschnitt 3.3 (S. 31) zu erstellen.

1. Markieren der zu extrahierenden Textabschnitte
   - Ermitteln und markieren, welche Abschnitte des Pattern-Formates Problem und Lösung beschreiben (z. B. [Tid05] bezeichnet „Problem“ als „Was“, „Warum“ und „Lösung“ als „Wie“)
   - Extrahieren von Quelle, Stimulus, Umgebung und Response-Prüfung aus der Problembeschreibung
   - Ermitteln von Response und Artefakt aus der Lösungsbeschreibung
2. Beschreiben des Interaktionsszenarios
   - Kombinieren der Textpassagen
   - Überarbeiten des Textes, damit dieser lesbarer und verständlicher wird
   - Referenzieren und kurzes Erläutern der Patterns, die Anforderungen des Interaktionsszenarios beschreiben
3. Klassifizieren des Interaktionsszenarios basierend auf Inhalten des Patterns
   - Vergeben einer Nummer und eines charakteristischen Namens
   - Zuordnen zu einer Interaktionskategorie
   - Zuordnen von Usability-Zielen, Begründung
   - Einschätzen der potenziellen Architektursensitivität basierend auf den Patterns und eigenen Erfahrungen, ergänzt um den Vertrauensfaktor
4. Evaluieren des Interaktionsszenarios durch einen Experten und anschließende Überarbeitung.

Bevor diese Extraktionsmethode am Beispiel gezeigt wird, betrachten die nächsten Abschnitte das Konzept der Patterns und die Auswahl der Pattern-Sammlungen.

3.4.2 *Patterns*

Das Konzept der Patterns (Muster) wurde von Alexander [AIS78] für ein Kompendium für Städte- und Gebäudearchitektur entworfen und etablierte sich zur Wissensspeicherung in der Softwareentwicklung, in der Mensch-Maschine-Interaktion (MMI) und in vielen anderen Fachbereichen. Ein Pattern ist ein Problem-Lösungs-Paar, das in einem bestimmten Kontext auftritt und durch ihn beeinflusst wird [AIS78]. Mit prägnanten Namen schaffen Patterns ein Vokabular für eine vereinfachte zwischenmenschliche Kommunikation unter ihren Anwendern. Um die Beschreibungen auf das Wesentliche zu reduzieren, sind die Muster in einem bestimmten Format verfasst. Dabei sollen neben dem Problem und der Lösung auch der Kontext, die Forces und die Konsequenzen erläutert werden [WF11, MD12]:

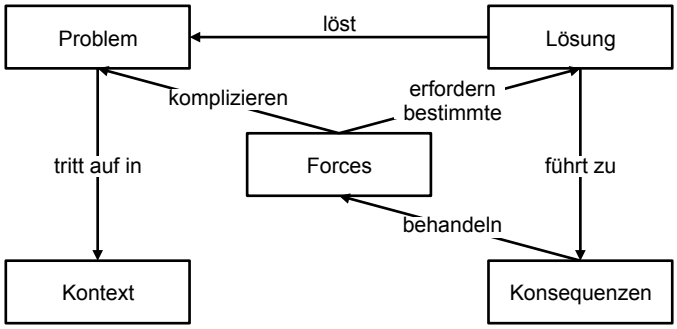

Abbildung 8: Zusammenhänge zwischen essentiellen Pattern-Abschnitten

- Kontext: beschreibt die unveränderlichen Rahmenbedingungen, in denen ein Pattern auftritt.
- Problem: beschreibt das Problem kurz und prägnant in einem Satz.
- Forces: listet auf, welche oftmals widersprüchlichen Anforderungen und Rahmenbedingungen (Kräfte) wirken, dass das Problem kompliziert zu lösen ist.
- Lösung: beschreibt die Lösung, meist zuerst in einem Satz, daraufhin detailliert.
- Konsequenzen: listet auf, inwiefern die Lösung auf Problem und Forces positiv, negativ oder gar nicht einwirkt.

Abbildung 8 illustriert die Zusammenhänge der essentiellen Pattern-Abschnitte. Die Lösung beschreibt den essentiellen Kern einer Idee zur Behebung eines speziellen Problems, das durch Forces verschärft wird. Die zu dieser Situation passende Lösung des Problems führt zu positiven und negativen Konsequenzen. Jede Force sollte durch eine Konsequenz aufgelöst werden und umgekehrt; nicht erfolgreich aufgelöste Forces führen zu weiteren Patterns. Die unveränderlichen Rahmenbedingungen bilden den Kontext, in dem das Problem auftritt.

Ein Architekturpattern bezieht sich auf ein Problem auf den Architekturebenen 1 und 2, betrifft also die allgemeine Systemstruktur und das Verhalten des Systems und nicht nur ein paar individuelle Subsysteme [BMR+96, AZ05].

Die vorhandene Literatur folgt zwei Denkschulen [AZ05]. Die erste verwendet den Begriff *Architekturpattern* [BMR+96, BRSS00, VKZ04] und die zweite den Begriff *Architekturstil* [SG96, SC97, BCK03, CKK02].

- Als *Architekturpatterns* werden Problem-Lösungs-Paare beschrieben, die in einem gegebenen Kontext auftreten und durch ihn beeinflusst werden (wie soeben erläutert). Architekturpatterns dieser Denkschule sind darauf ausgelegt, in einer *Pattern Language* durch Abhängigkeiten und Beziehungen untereinander verknüpft zu werden. Das Ideal, aus philosophischer Sicht, ist die „zeitlose“ Qualität eines Patterns [AIS78].

- *Architekturstile* fokussieren Komponenten, Konnektoren und den Daten- und Kontrollfluss [SC97, BCK03], sowie eine Menge von Rahmenbedingungen [BCK03, CKK02] und Konfigurationen [MM03]. Sie sind sehr konkret.

  Im Gegenteil zu Architekturpatterns werden das Problem und die Gründe zur Auswahl einer Lösung nicht ausführlich betrachtet. Diese Beschreibung führt einerseits dazu, dass es von einem Pattern viele Variationen gibt. Andererseits vereinfachen die genauen Implementierungsdetails den Einsatz von Architekturbeschreibungssprachen.

Architekturpatterns und Architekturstile sind prinzipiell die gleichen Konzepte und unterscheiden sich insbesondere durch verschiedene Beschreibungsformate. Deshalb wird für beide Perspektiven auf Patterns der Begriff Architekturpattern verwendet.

3.4.3 *Ausgewählte Pattern-Sammlungen*

Die Interaktionsszenarios beschreiben Interaktionen, die (1) gültig sind, weil sie in der Praxis auftreten, die (2) möglicherweise architektonisch unterstützt werden müssen, falls sie noch nicht in der Softwarearchitektur berücksichtigt wurden, und die (3) für mobile Anwendungen relevant sind. Die Patterns sollen folgenden Ansprüchen genügen:

1. Praxisbezug: Eine häufige Verwendung spricht dafür, dass ein Pattern häufig in der Praxis auftritt.

2. Potenziell architektursensitive Interaktion: Ein Pattern thematisiert eine Interaktion und/oder die technische Umsetzung von Usability-Funktionen oder konkrete Zusammenhänge zwischen Usability und Softwarearchitektur.

3. Relevanz für mobile Anwendungen: Ein Pattern ist speziell für mobile Anwendungen beschrieben und/oder schließt mobile Anwendungen nicht grundsätzlich aus. Ausgeschlossen wäre beispielsweise ein Pattern, dass sich auf die Darstellung auf großen Bildschirmen bezieht und damit grundsätzlich nicht für die kleineren Bildschirme mobiler Endgeräte geeignet ist.

Aus einer Vielzahl von Pattern-Sammlungen wurden folgende Quellen ausgewählt:

- Patterns über den Zusammenhang zwischen Usability und Softwarearchitektur:
  - „Usability Patterns“ und „Bridging Patterns“ von Folmer et al. [FGB03, FvWB06]: vier komplette und 15 abstrakt gehaltene Architekturpatterns,
  - „Architecture-sensitive Usability Patterns“ von Bass et al. [BJK01] und Golden et al. [GJB05]: beschreiben 26 „Usability-Szenarios“ und detaillierte Architekturstile über Usability und ihre architektonische Unterstützung,

- Interaktionen, die potenziell architektursensitiv sein können:
  - Interaktionsdesign-Patterns von Tidwell [Tid05]: umfassende Sammlung von Mustern zu einer Vielzahl von technisch zu unterstützenden Interaktionen – 75 von 94 Patterns, d. h. nur Patterns zum Interaktionsdesign, keine zu den allgemein gehaltenen Prinzipien und keine über Ästhetik.

- Patterns über Mobilität:
  - „Interface-Design-Patterns“ von Little Springs Design [Lit10]: 32 unterschiedlich vollständige Patterns, insbesondere für mobile Anwendungen.

3.4.4 *Beispiel*

Tabelle 12 zeigt, wie aus dem Usability-Patterns Cancelability [Tid05] Textabschnitte für das Interaktionsszenario „Cancel“ extrahiert wurden. In den Patterns werden Textabschnitte, welche in den jeweiligen Quellen die Interaktion (1), den Stimulus (2), die Response (3) und die Response-Prüfung (4) beschreiben, markiert und dann extrahiert. Anschließend werden sie klassifiziert, d.h. einer oder mehreren Interaktionskategorien zugeordnet und basierend auf den jeweiligen Patterns werden ihre Usability-Ziele und der mögliche Änderungsaufwand ergänzt.

Pattern-Abschnitt	*Extraktion*
„What“	„Provide a way to *instantly cancel*[3,4] a time-consuming *operation*[2], with *no side effects*[4]“
„Why“	„*Users*[1] *change their minds*[2]. Once a time-consuming *operation starts*[2], a user may *want to stop it*[2], especially if a Progress Indicator tells her that it'll take a while. Or the *user*[1] may *have started it*[2] by accident in the first place. Cancelability certainly helps with error prevention and recovery – a user can cancel out of something she knows will fail, like loading a page from a web server she realizes is down. (...)“
„How“	„(...) Put a *cancel button*[3] directly on the interface, next to the Progress Indicator (...) or wherever the results of the operation appear. (...) When the *user*[1] *clicks or presses the Cancel button*[2], *cancel the operation*[3] *immediately*[4]. If you wait too long, for more than a second or two the user may doubt, that the cancel actually worked (or you may dissuade him from using it, since he might as well wait for the operation to finish). *Tell the user that the cancel worked*[4] – halt the progress indicator, and show a status message on the interface, for instance. (...)“
[1] Quelle, [2] Stimulus, [3] Response, [4] Response-Prüfung	

Tabelle 12: Extraktion der Beschreibung des Interaktionsszenarios Cancel (Abbrechen) aus dem Pattern Cancel [Tid05]

Tabelle 13 zeigt das anhand der Patterns erstellte und textuell überarbeitete Interaktionsszenario Nr. 4 *Abbrechen*, das auf den Patterns Cancelability [Tid05], Cancel [FGB03] und Canceling Commands [BJK01] basiert.

Name	*Beschreibung*
Nummer	4
Kurzname	Abbrechen (Cancel)
Quelle	Benutzer
Stimulus	haben eine Operation gestartet und möchten sie stoppen und widerrufen.
Response	Das System beendet die Ausführung, sobald der Nutzer die Abbrechen-Funktion aktiviert. Der vor dem Starten des Vorgangs bestehende Systemstatus wird wieder hergestellt.
Response-Prüfung	Das Abbrechen wird innerhalb einer bestimmten Zeitspanne durchgeführt. (A, U) Der Prozess des Abbrechens wird gegenüber dem Benutzer kommuniziert. (A, U)
Usability-Pattern(s)	[Tid05] 50. Cancelability (Abbruchmöglichkeit): Biete eine Möglichkeit an, um eine zeitaufwendige Aktion abzubrechen, ohne dass Nebenwirkungen auftreten. [BJK01, GJB05] 3. Canceling Commands: Ein Nutzer ruft eine Funktion auf und möchte dann, dass die Funktion nicht länger ausgeführt wird. Systeme sollten dem Nutzer erlauben, Abläufe abzubrechen. [FGB03] 19. Cancel: Erlaube dem Nutzer beispielsweise zur Verhinderung von Fehlern, einen Befehl abzubrechen, der aufgerufen, aber noch nicht abgeschlossen wurde.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effizienz: Nach einem Fehler ist das System schnell wieder einsatzfähig. (+) Effektivität: Das System arbeitet zuverlässig und ermöglicht die Weiterbearbeitung der Aufgabe des Benutzers. (+) Sicherheit: Die Daten der Benutzer sind sicher (die Daten, mit denen sie gearbeitet haben, Arbeitsergebnisse). (+) Erlernbarkeit: Lernen durch Versuch und Irrtum möglich.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 13: Interaktionsszenario Abbrechen (Cancel)

Die Vorlage für ein Interaktionsszenario befindet sich im Anhang, Abschnitt A.1, in Tabelle 61 auf S. 203.

Im Anhang werden 50 Interaktionsszenarios aufgeführt (Abschnitt A.2, S. 204 ff.; siehe auch http://cassini.saturn-lab.com/). Die Datenbankanwendung setzt HTML 5.0 und CSS ein und kommuniziert über PHP 5.0 mit einer Datenbank auf einem MySQL-5.1-Server (Screenshot siehe Abbildung 9).

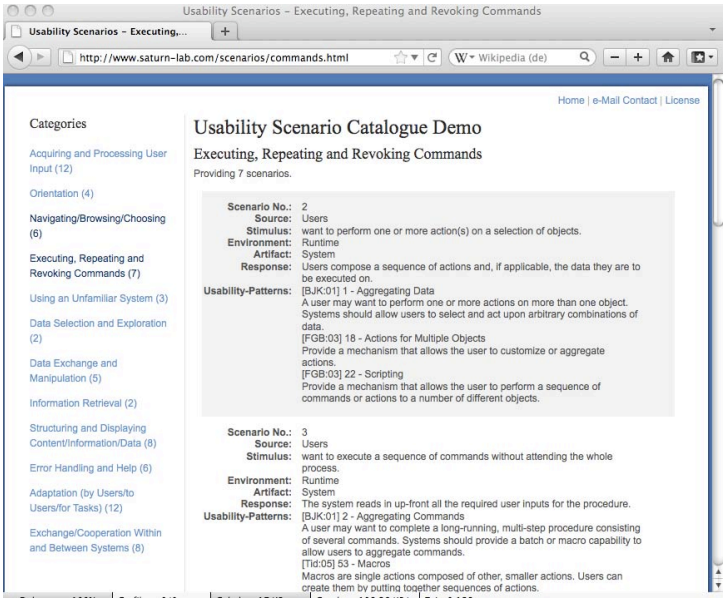

Abbildung 9: Screenshot des Katalogs der Interaktionsszenarios (letzter Aufruf 16.05.2011)

Aufgeführte Patterns sind referenziert; eine digitale Pattern-Library kann aus Urheberrechtsgründen nicht zugänglich gemacht werden. Die Datenbankstruktur des Interaktionsszenariokataloges zeigt Abbildung 10 als UML-Klassendiagramm, das repräsentiert, wie die Elemente eines Interaktionsszenarios zusammenhängen.

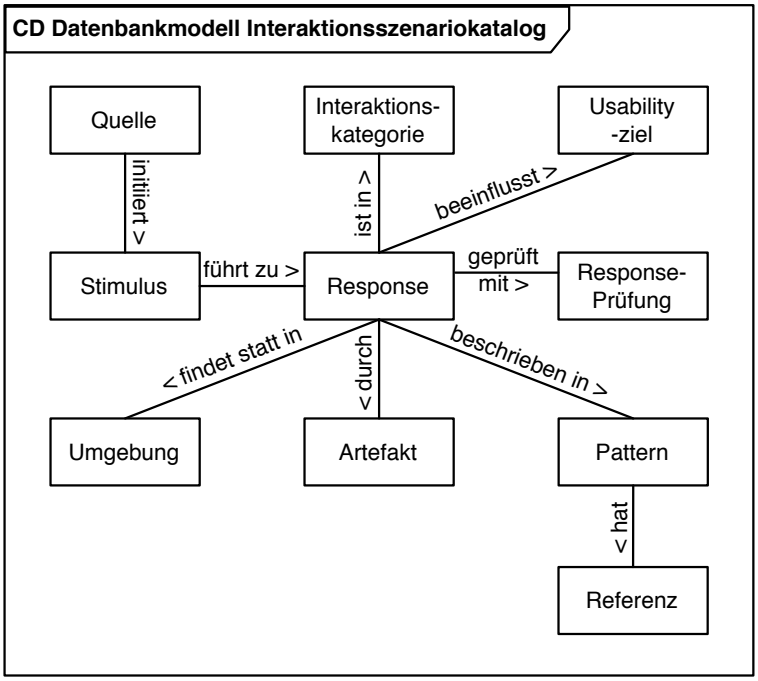

Abbildung 10: Datenbankmodell Interaktionsszenariokatalog

Während und nach der Methode SATURN unterliegen die Interaktionsszenarios dem in Abbildung 11 dargestellten Interaktionszenario-Lebenszyklus.

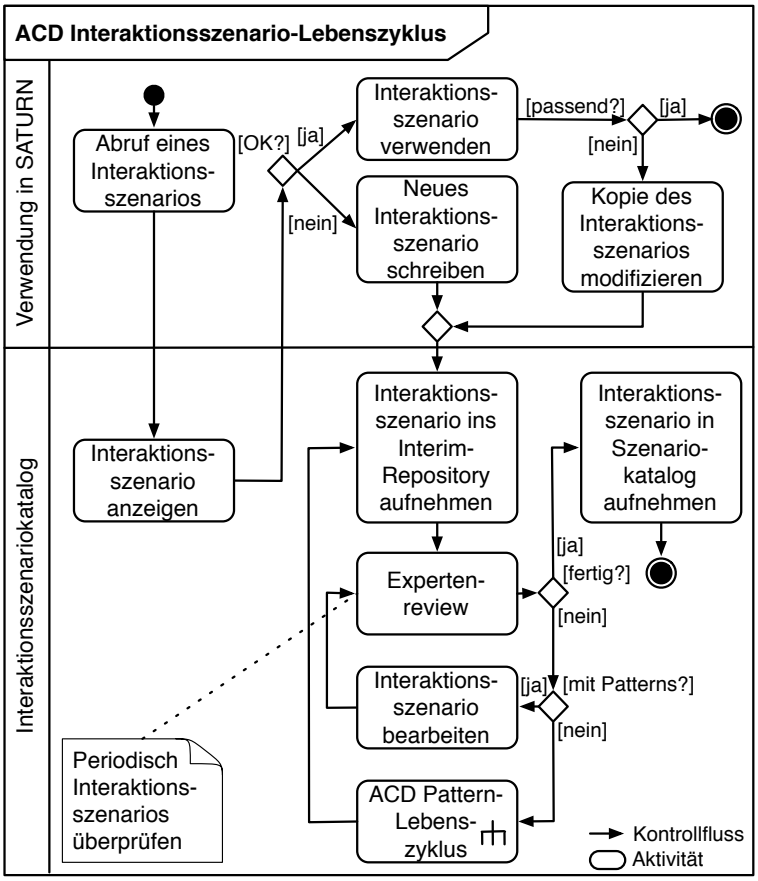

Abbildung 11: Aktivitätsdiagramm Interaktionszenario-Lebenszyklus

Wenn ein Interaktionsszenario ausgewählt wurde, wird es für die Analyse angepasst. Dieses fließt in das Interim-Repository ein, wo es von einem oder mehreren Experten überprüft wird. Es wird überarbeitet, angepasst oder verworfen. Falls noch kein Pattern für ein Interaktionsszenario existieren sollte (das ist nach einem Analysedurchgang durchaus möglich), erfordert das Vorgehen zur Erstellung gültiger Interaktionsszenarios als Grundlage ein oder mehrere gültige Patterns. Wenn diese nicht vorliegen oder verfügbar sind, müssen eigene Patterns erstellt, evaluiert, verwaltet und gepflegt werden.

Es ist auch möglich und üblich, Patterns als gedruckte Dokumente zu verwalten. Für Entwicklungsteams oder Organisationen bietet es sich zur Fokussierung und Anpassung an eigene Kontexte an, interne Pattern-Sammlungen zu erstellen und zu pflegen. Damit können genau die Lösungen dokumentiert werden, die in den eigenen Systemen relevant sind. Wird eine Software zur Unterstützung gewünscht, ist der Einsatz einer Pattern-Bibliothek [Tid05] hilfreich.

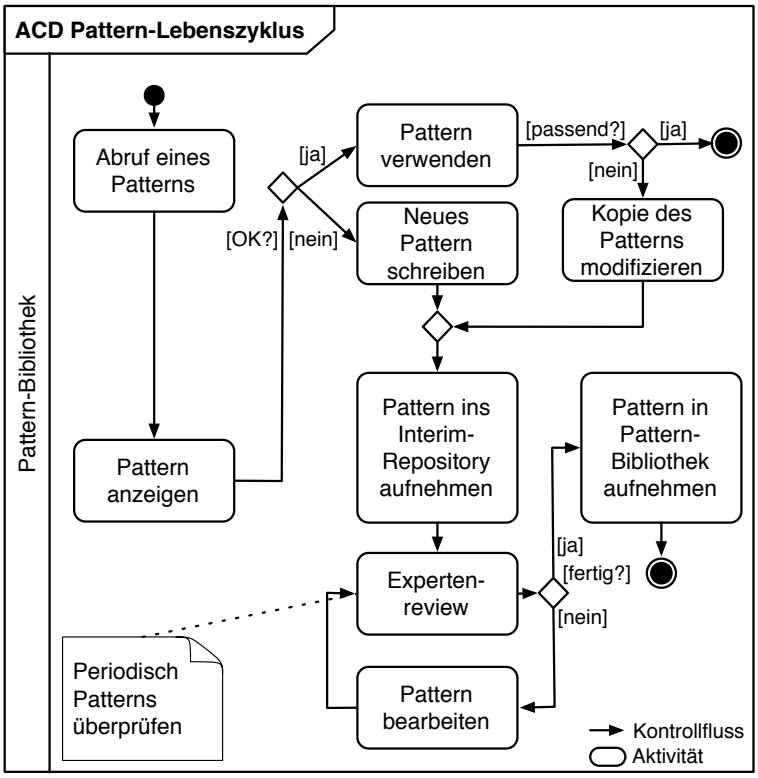

Abbildung 12: Aktivitätsdiagramm Pattern-Lebenszyklus

Unabhängig von einer textuellen oder softwaregestützten Verwaltung von Patterns wird sich zur Qualitätssicherung eines Patterns der in Abbildung 12 dargestellte Pattern-Lebenszyklus festgelegt [GBG09]. Identifizierte Lösungen werden als Pattern beschrieben und durch Experten dahingehend evaluiert, ob sie eine gute Design-Lösung für ein Problem darstellen. Eine gute Lösung wird in ein Interim-Repository übernommen. Kommt die Lösung in verschiedenen Projekten zum Einsatz, wird sie erneut durch Experten evaluiert, als Pattern mit Qualitätsindex versehen und in die Bibliothek übernommen. Dieser Qualitätsindex entspricht dem bei den Interaktionsszenarios eingesetzten Vertrauensfaktor (Tabelle 10, S. 37).

3.6 ZUSAMMENFASSUNG

Eine Wissensbasis von Interaktionsszenarios für SATURN wurde entworfen und erstellt, um erstens, während der Softwarearchitektur-Analyse die Ressourcen zu sparen, die notwendig wären, wenn bei jeder Analyse neue Interaktionsszenarios beschrieben werden müssten, und zweitens, um eine gute Qualität der verwendeten Interaktionsszenarios zu garantieren und um die Interaktionsszenarios vorzubereiten, die potenziell architektursensitiv sind.

Die Interaktionsszenarios basieren auf [CKK02, BCK03], unterscheiden sich aber grundsätzlich: sie sind nicht im Kontext Qualitätsattribute angesiedelt, sondern im Kontext von Anforderungen. In diesem Kapitel wurden ihre Kernelemente und ihre Klassifikation definiert. Ein bekannter Ansatz zum Extrahieren von Szenarios aus Patterns [ZBJ04] wurde auf Interaktionspatterns angewandt. Patterns sind ein etabliertes Mittel zur Dokumentation von Best Practices zu wiederkehrenden Herausforderungen, die hier als Anforderungen verstanden werden. Somit werden Interaktionsszenarios beschrieben, die potenziell architektursensitive Usability-Anforderungen darstellen.

Damit die Gültigkeit der Interaktionsszenarios gewährleistet werden kann und die Wissensbasis gepflegt wird, wurden sowohl ein Interaktionsszenario- als auch ein

Pattern-Lebenszyklus aufgestellt, mit dem die Qualität der Interaktionsszenarios und Patterns in einer schriftlichen oder in einer software-gestützten Pattern-Bibliothek gesichert werden kann.

Der Katalog der Interaktionsszenarios liegt als Text im Anhang (Abschnitt A.2, S. 204 ff.) vor, die aktuelle Version ist über die Website http://cassini.saturn-lab.com/ verfügbar.

Mit diesem Kapitel wurden somit folgende Forschungsfragen beantwortet: es wurde ein geeignetes Format für Szenarios gefunden (Forschungsfrage 2 beantwortet), es wurde dargestellt, wie andere Forscher das Qualitätsmerkmal Usability in Szenarios operationalisieren (Forschungsfrage 3 thematisiert). Die Interaktionsszenarios wurden klassifiziert, um ihre Auswahl zu erleichtern (nützlich im Hinblick auf Forschungsfrage 4). Insgesamt bilden diese Interaktionsszenarios den ersten Schritt dahin, die neue Softwarearchitekturanalysemethode so zu gestalten, dass sich die Anwender dieser Methode auf ihre Anforderungen konzentrieren können, so dass die Methode SATURN unabhängig von einem hohen Wissen über Patterns und Best Practices wird (Forschungsfrage 5). Das folgende Kapitel behandelt weitere Forschungsfragen zur Erstellung der Methode SATURN und stellt sie schließlich vor.

DANKSAGUNGEN

Eine vorläufige Version der Abschnitte dieses Kapitels zu Usability-Definitionen, den verwandten Arbeiten, der Erhebung, Definition und Klassifikationen (Interaktionskategorien, Usability-Ziele, potenzielle Architektursensitivität) sowie ein erster Katalog der Interaktionsszenarios erschienen 2009 in der Diplomarbeit von Stefan Seitz [Sei09], die im Rahmen dieser Forschungsarbeit erstellt wurde. Diese Ergebnisse wurden auf der *MobiWIS 2011* mit Stefan Seitz und Volker Gruhn veröffentlicht [BSG11]. Katharina König übersetzte die erste Version der Interaktionsszenarios vom Englischen ins Deutsche. Der Abschnitt dieses Kapitels über den Pattern-Lebenszyklus erschien im Journal *it – Information Technology* 2009 mit Thomas Grill und Volker Gruhn [GBG09]. Michael Seiltz erforschte in seiner Bachelorarbeit, wie eine Pattern-Bibliothek in einem Content Management System umgesetzt werden kann [Sei10]. Der Vertrauensfaktor wurde im Workshop *Trustworthy Ubiquitous Computing* vorgestellt und im Rahmen der Konferenz *Mobile Multimedia 2008* mit Thomas Grill und Volker Gruhn veröffentlicht [BGG08] (Best Workshops Paper Award).

4 DIE ERSTELLUNG DER METHODE SATURN

Um die neue Methode zu erstellen, behandelt Abschnitt 4.1 die Forschungsfragen dieser Arbeit. Als Ergebnis davon wird die Methode SATURN in Abschnitt 4.2 präsentiert, gefolgt von einer ergänzenden Methode, die beschreibt, wie diese Softwarearchitekturanalysemethode mit mit einem Nutzertest kombiniert werden kann (Abschnitt 4.3). In Abschnitt 4.4 wird rückblickend geprüft, inwiefern die Forschungsfragen beantwortet wurden. Eine abschließende Zusammenfassung enthält Abschnitt 4.5.

4.1 ERSTELLEN DER METHODE SATURN

4.1.1 *Struktur und Bestandteile einer Methode*

Mit Architekturanalysen werden Softwarearchitekturen dahingehend untersucht, ob sie bestimmte Qualitätsmerkmale unterstützen und ob notwendige Änderungen besonders aufwendig sind [DN02]. Die Erstellung und die Analyse einer Softwarearchitektur führen zu konkreteren oder veränderten Anforderungen [BCK03], sowie zu neuen Anforderungen, die nicht auf Kundenwünschen, sondern auf Erkenntnissen aus der Architekturanalyse basieren [HHP00].

Softwarearchitekturanalysemethoden nutzen Metriken, Fragetechniken oder beides [CKK02, BCK03]. Frühere Methoden untersuchen die architektonische Unterstützung von Usability mit Fragetechniken (Kapitel 2, S. 8ff.). Dies ist sinnvoll, da Usability als Qualität der Benutzung sonst nur mit Hilfe des User Interfaces mit Benutzern oder Experten evaluiert werden kann. Die neue Methode arbeitet deshalb ebenso mit Fragetechniken und wird als szenario-basierte Softwarearchitekturanalysemethode konzipiert. Typische Arbeitsschritte nach [DN02, BG04] sind das Definieren der Ziele, das Operationalisieren der Anforderungen eines Qualitätsmerkmals in Szenarios, das Evaluieren deren Umsetzung in der Softwarearchitektur und das Interpretieren der Ergebnisse.

Typische Elemente von Methoden sind Aktivitäten, Rollen, Spezifikationsdokumente, Techniken, Werkzeuge und ein Metamodell. Aufeinander folgende Aktivitäten bilden ein Vorgehensmodell; Aktivitäten selbst sind Konstruktionsaufgaben, die Rolleninhaber ausführen, um definierte und strukturierte Spezifikationsdokumente zu erstellen. Dafür werden Techniken genutzt, d. h. detaillierte Instruktionen. Werkzeuge können, aber müssen nicht, die Anwendung einer oder mehrerer Aktivitäten unterstützen. Ein Metamodell stellt die theoretischen Elemente und ihre Zusammenhänge als Grundgerüst einer Methode dar. [Gut94, BWHW05]

Die folgenden Abschnitte behandeln die aufgestellten Forschungsfragen (Kapitel 2, S. 8ff.).

4.1.2 *Forschungsfrage 1: Wie und wann mobilen Nutzungskontext integrieren?*

Mobilität bedeutet Beweglichkeit [WKRSS09] und beschreibt im Rahmen dieser Arbeit die passive Eigenschaft, dass Personen, Gegenstände oder Logik beweglich sein können. Für Benutzer heißt mobil sein, dass sich der Ort des Benutzers, der physikalische und der soziale Kontext ändern können und dass sie sich nicht auf physikalische Ressourcen verlassen können [Bal07]. In Bezug auf Informationssysteme unterscheidet Antoniac [Ant05] die Mobilität von Menschen, Informationen und Infrastrukturen. Analog dazu nennt Pandya [Pan99] aus technologischer Sicht die Gerätemobilität, bei

der ein Gerät physikalisch bewegt wird, die Benutzermobilität, bei der ein Benutzer verschiedene Geräte verwenden kann, um auf einen Dienst zuzugreifen, und die Dienstmobilität, bei der auf einen Service von überall her zugegriffen werden kann.

Roman et al. betrachten schließlich physikalische Einheiten wie Personen und Geräte sowie logische Einheiten wie Dienste oder Rechenkomponenten. Sie verstehen Mobilität als die Art, mit der logische oder physikalische Komponenten ihre Lokation ändern können. Damit sind Kontext, Kontextänderungen und die Reaktion des Systems darauf bei der Entwicklung mobiler Anwendungen wesentlich. [RPM00]

Die Rahmenbedingungen mobiler Anwendungen, die Usability beeinflussen, werden durch den Nutzungskontext mobiler Anwendungen (mobiler Nutzungskontext) definiert. Dieser beschreibt die komplexen Abhängigkeiten eines mobilen interaktiven Systems und umfasst die fünf Kontextfaktoren Benutzer, Technologie, Umwelt, Aufgaben und die mobile Anwendung selbst.

Coursaris und Kim [CK06] präsentieren ein Rahmenwerk der Kontextfaktoren Umwelt, Nutzer, Aufgabe und Technologie. Dieses basiert auf einer Analyse von empirischen Usability-Studien hinsichtlich der Verwendung von mobilen Anwendungen. Die Forscher argumentieren, dass die Resultate einer Usability-Evaluation von einer ausdrücklichen Betonung des Kontexts und der Diskussion der Technologie jenseits der Benutzerschnittstelle besonders profitieren können. Sie beziehen sich insbesondere auf technologische Aspekte wie Netzwerkverbindung, Verlässlichkeit und Einfluss auf die Speicherfähigkeiten eines mobilen Endgerätes.

Tamminen [TOTK04] präsentiert mobile Kontextparameter, welche die Nutzung einer mobilen Applikation beeinflussen. Carter und Mankoff [CM04] thematisieren Herausforderungen bei der Durchführung der Usability-Evaluation in ubiquitären Umgebungen. Auf Basis des Kontexts identifizierten sie Parameter, die besonders schwierig zu handhaben sind und die deshalb besonders sorgfältig während des Designs einer Evaluation beachtet werden müssen: die verwendeten Metriken, die Skalierbarkeit eines Szenarios, die Mehrdeutigkeit gesammelter Daten und die Unaufdringlichkeit eines Systems, das im täglichen Leben verwendet werden soll. Hummel und Kollegen [HHG08, TGHW06] erforschten Parameter und deren Einfluss auf die Benutzung von mobilen Anwendungen in der realen Welt.

Basierend auf diesen Forschungsergebnissen und den Ergebnissen von Schmidt [SBG99], Tarasewich [Tar03] sowie jene von Varshney und Vetter [VV02] wurden die Kontextfaktoren Umgebung, Benutzer, Aufgaben, Geräte und Anwendungen identifiziert. Diese Kontextfaktoren beschreiben anhand von Literaturstudien erstellte Profile, deren Parameter in Tabellen zusammengefasst werden.

Abhängig von den Anforderungen einer Anwendung muss geprüft werden, ob ein einzelner Parameter für das Design und die Evaluation des Systems beachtet werden muss oder nicht. SATURN leitet die Teilnehmer der Analyse mit den folgenden Checklisten zur Erstellung der Kontextfaktoren an. Es sind meist mehrere Benutzer-, Umgebungs- und Gerätetypen zu spezifizieren, dazu ein kleines Set von Aufgaben und grundlegende Merkmale der Anwendung.

KONTEXTFAKTOR UMGEBUNG Der Begriff Umgebung oder Umwelt wird bestimmt durch die Verhältnisse, Objekte oder Bedingungen, durch die jemand oder etwas umgeben ist[1]. Parameter ermöglichen eine Momentaufnahme durch bestimmte Informationen über die Umgebung, beispielsweise Sensordaten über Wetterkonditionen (physikalischer Kontext), ein privates oder geschäftliches Treffen (sozialer Kontext). Solche Informationen können anhand historischer und in diesem Moment wahrgenommenen Daten ausgewertet werden. Gemeinsam können sie die aktuelle Situation deuten oder ähnliche Situationen und Reaktionen darauf vorhersagen.

1 Merriam-Webster Online Dictionary, 18. April 2012, `http://www.merriam-webster.com/dictionary/environment`

Name	*Beschreibung*
Ort und Orientierung	Geographische Daten zur Auswertung der Bewegung und des Ortes des Nutzers, Starten von Interaktionen basierend auf Lokationsänderungen
Physische Eigenschaften	Umgebungsdaten wie Temperatur, Geräusche, Feuchtigkeit, Licht
Soziale Bedingungen	Co-Lokation, Gruppendynamik, Beziehung der Teilnehmer zur Umgebung, Aktivität (Freizeit, Arbeit, Party, Meeting)
Konnektivität	Verbindungsart, erwartete Abdeckung, Stabilität
Kollaborationen	Fähigkeit, mit anderen Geräten zu kollaborieren

Tabelle 14: Kontextfaktor Umgebung

KONTEXTFAKTOR BENUTZER Im Rahmen der Produktentwicklung des Marketings werden Zielgruppen segmentiert [KK08], auf deren Bedürfnisse Produkt, Distribution, Preispolitik und Promotionsaktivitäten abgestimmt werden. Analog dazu beschreibt der *Kontextfaktor Benutzer* Benutzerprofile, die sich zur Auswahl von Benutzergruppen für Evaluationen und zur Erstellung von Personas eignen. Personas beschreiben repräsentative, aber fiktive Benutzer einer Anwendung [GP02, Coo95]. In benutzerzentrierten Designprozessen werden Anforderungen mittels Benutzerprofilen und Personas analysiert und beschrieben [CRC07, Hai07, GP02, PG03].

Das Benutzerprofil enthält Eigenschaften wie demografische Daten, Charakterzüge, Intelligenz, arbeits- oder aufgabenbezogene Faktoren, den Expertenstatus (Neuling, Erfahrene, Experte), kulturelle Dimensionen (Sprache, Herkunft), Arbeitsrollen (z.B. Ärzte, Ingenieure), Behinderungen, Alter, Geschlecht und Bewegung [ZCTT05, CK07].

Name	*Beschreibung*
Interaktionsanforderungen der Benutzer	
Aufmerksamkeitsspanne	Fähigkeiten/Behinderungen bzgl. Aufmerksamkeit (Richtwert: 5 bis 8 Sek. für eine Interaktion, bis 20 Sek. für eine Aufgabe [Oul05])
Motorische Fähigkeiten	Fähigkeiten/Behinderungen bzgl. Interaktion mit Hilfe des menschlichen Körpers
Audiovisuelle Fähigkeiten	Fähigkeiten/Behinderungen bzgl. des Hörens und Sehens
Mentale Fähigkeiten	Fähigkeiten/Behinderungen bzgl. des kognitiven Aufwandes
Expertenstatus	Fähigkeiten aufgrund der Erfahrung (Neulinge, Erfahrene, Experten)
Kultur	Kulturelle Besonderheiten (Sprache, Farbverständnis, Verhaltensregeln)
Demografie	Demografische Daten (z. B. Altersspanne, Geschlecht)
Arbeitsrollen	Beruf, Arbeitsinhalt (z. B. Ärztin, Ingenieur, Professor)
Nutzungsverhalten	
Bevorzugte Lokation	Benutzung der Anwendung bspw. im Büro, im Freien, in öffentlichen Verkehrsmitteln (siehe Kontextfaktor Umgebung)
Verwendung von Multimedia	Nutzung von Audio, Video, VOIP, Triple Play
Nutzungshäufigkeit	Frequenz der Verwendung der Anwendung

Tabelle 15: Kontextfaktor Benutzer

KONTEXTFAKTOR AUFGABE Aufgaben können auch als Anwendungsfälle visualisiert werden [Coco1]. Kategorisiert werden sie anhand der Parameter in Tabelle 16.

Name	*Beschreibung*
Funktionalität	Funktionalität durch Ausführung einer speziellen Aufgabe
Workflow	Kombination von Aufgaben zur Durchführung von Aufgaben mit komplexeren Funktionalitäten
Interaktionen	Mögliche Interaktionen zum Durchführen einer Aufgabe
Dauer	Minimale und maximale Dauer der Aufgabe
Komplexität	Interaktivität zwischen Mensch und Computer (Wechselbeziehungen)
Abhängigkeiten	Abhängigkeiten von externen Parametern, z.B. Aufgabe nicht bei Bewegung durchführbar

Tabelle 16: Kontextfaktor Aufgabe

KONTEXTFAKTOR ENDGERÄT Mobile Endgeräte unterscheiden sich von stationären Geräten durch die Rechnungsleistung, die Speicherressourcen, die Eingabe- und Ausgabemöglichkeiten sowie durch die Netzwerkmöglichkeiten [HTKR05]. Da sie sich auch untereinander stark unterscheiden, werden Endgeräteprofile zu ihrer Kategorisierung genutzt. Ein Beispiel für solche online verfügbaren Device Profiles ist das Open Source „Wireless Universal Resource File“ (WURFL)[2]. Die Entscheidung, welche Klassen von Endgeräten unterstützt werden und damit, welche Restriktionen beachtet werden müssen, beeinflusst in beträchtlichem Maße Design und Implementierung einer Anwendung und muss deshalb frühzeitig im Entwicklungsprozess getroffen werden. Tabelle 17 zeigt Parameter, die auf Literaturstudien basieren [Bal07, JM06, SHKB05, CI06, CN08, GBG09].

Name	*Beschreibung*
Gewicht	Gewicht der Endgeräte (Angaben der Art „leicht“, „schwer“)
Größe des Gerätes	Maße des Endgerätes (bspw. Länge, Breite, Dicke)
Robustheit	Anforderungen an Stabilität der Hardware
Bildschirmgröße	Unterstützte Bildschirmgrößen
E/A-Möglichkeiten	Interaktionsmöglichkeiten durch Hardware und Software
Betriebssystem	Mobile Plattform (bestimmt Funktionsumfang, Versionen von Anwendungen, bspw. von Browser, Fähigkeit für Multitasking)
Stromverbrauch	Spezielle Anforderungen an Akku-Laufzeit
CPU-Performanz	Spezielle Anforderungen an CPU-Leistung (alte Geräte mit geringer CPU erfordern Reduktion von Komplexität von Berechnungen)

Tabelle 17: Kontextfaktor Endgerät

2 WURFL, 28. Februar 2014, `http://wurfl.sourceforge.net/`

KONTEXTFAKTOR MOBILE ANWENDUNG Roman et al. [RPM00] thematisieren wesentliche Konzepte zur Beschreibung mobiler Anwendungen: deren Verwendung an einem oder mehreren Orten (Räume), die Art der physikalischen und logischen mobilen Elemente, Kontext, Kontextänderungen und deren Konsequenzen. Auch unvorhersehbare Kontexte und Kontextänderungen müssen in das Design der Anwendung und des Interfaces einbezogen werden [TGHW06].

Entsprechende Parameter des Kontextfaktors Anwendung werden in der Tabelle 18 aufgezeigt. Dieser Überblick über wesentliche Eigenschaften einer mobilen Anwendung wird durch eine genauere Beschreibung der Softwarearchitektur ergänzt.

Name	*Beschreibung*
Räume	Umgebungen, in denen die Anwendung verwendet wird (Referenz auf Kontextfaktor Umgebungen)
Physikalische Einheiten	Endgeräte und beteiligte Systeme zur Kollaboration (Referenzen auf Kontextfaktoren Endgeräte und Umgebung)
Kontext	Definition von Kontext, Context-Awareness und Konsequenzen bei Kontextänderungen (z. B. Lokation)
Logik	Verteilung der Logik zwischen mobilen und ggf. stationären Einheiten (Client, Proxy, Server)
Mobile logische Einheit	Bestimmung der über das Netzwerk drahtlos übermittelten Daten
Datenhaltung	Verteilung der Datenhaltung (Client, Proxy, Server)
Kommunikation	Geschätzter Kommunikationsaufwand innerhalb des Systems, bspw. zwischen Client und Server
Netzwerk	Anforderungen an Netzwerkkonnektivität durch Kommunikationsaufwand (z. B. Client-Server)

Tabelle 18: Kontextfaktor Mobile Anwendung

Damit auch Personen, die keine Usability-Experten sind, den Nutzungskontext schnell ermitteln können, soll er als Checkliste integriert werden. Damit von ihm konkrete Szenarios abgeleitet werden können, sollen ermittelte Rahmenbedingungen und funktionale Anforderungen mit dem Katalog der Interaktionsszenarios verbunden werden. Dazu soll eine Person, welche die Analyse durchführt, die 50 Interkationsszenarios bereits kennen.

4.1.3 *Forschungsfragen 2 und 3: Welches Format für Szenarios und wie Usability operationalisieren?*

Formate von Szenarios sowie Möglichkeiten und Grenzen dazu, Szenarios bezüglich Usability durch Personen erheben zu lassen, die keine Usability-Experten sind, wurden in den Abschnitten 3.2.1 (S. 27) und 3.2.2 (S. 30) betrachtet. Daraufhin wurde das Format der Interaktionsszenarios in Abschnitt 3.3 (S. 31) definiert. Wie sie erhoben werden, beschreibt Abschnitt 3.4 (S. 39ff.).

In der Methode SATURN sollen diese Interaktionsszenarios verwendet werden. Allerdings erfordern die Qualitätsanforderungen an die Interaktionsszenarios des Kataloges, dass jedes Interaktionsszenario von einem oder mehreren Patterns abgeleitet ist. Der damit verbundene Arbeitsaufwand, ein komplettes Interaktionsszenario zu erstellen, würde den Rahmen der Analyse sowohl inhaltlich als auch zeitlich sprengen. In Abschnitt 3.5 (S. 44ff.) wird deshalb der Lebenszyklus eines Interaktionsszenarios be-

schrieben. Dieser verdeutlicht, dass in der Methode neu erstellte Interaktionsszenarios in ein Interim-Repository aufgenommen und später weiter bearbeitet werden.

Die Methode soll dazu beitragen, dass dieser Lebenszyklus eingehalten wird. Es soll also ein Rückblick aufgenommen werden, der auch die Überarbeitung der Interaktionsszenarios thematisiert.

4.1.4 *Forschungsfrage 4: Wie Szenarios fachlich motiviert auswählen?*

Die Auswahl der Interaktionsszenarios soll fachlich motiviert erfolgen; bezüglich Usability eignet sich der Nutzungskontext als fachliche Grundlage, da er die Rahmenbedingungen für die Nutzung bestimmt. Rahmenbedingungen für den wirtschaftlichen Erfolg einer mobilen Anwendung erscheinen ebenfalls besonders sinnvoll, daher sollen hier auch wirtschaftliche Aspekte betrachtet werden.

Mobile Anwendungen betreffen (meist) einen konkreten Anwendungsfall, wie beispielsweise e-Mails bearbeiten oder Kontakte verwalten. Dieser Hauptanwendungsfall ist der Grund dafür, dass sich Käufer für eine Anwendung interessieren; dieser muss also auf jeden Fall funktionieren und sehr gut benutzbar sein. Allerdings gibt es viele ähnliche Produkte am Markt, wobei das Alleinstellungsmerkmal eine mobile Anwendung gegenüber ihrer Konkurrenz auszeichnet (beispielsweise ein bestimmtes Feature). Aus diesen Gründen sollen Interaktionsszenarios, die dem Hauptanwendungsfall oder dem Alleinstellungsmerkmal einer mobilen Anwendung zugeordnet werden können, immer für die Analyse ausgewählt werden.

Die Methode soll außerdem so aufgebaut sein, dass sie ausdrücklich fachliche Begründungen abfragt. Es soll festgehalten werden, welche Stakeholder ein bestimmtes Interaktionsszenario befürworten. Somit sind Meinungen von Stakeholdern auch in den nachfolgenden Arbeitsschritten sichtbar.

4.1.5 *Forschungsfrage 5: Hilfsmittel ohne Pattern-Expertise?*

Frühere Methoden erfordern es, Muster (Patterns, Abschnitt 3.4.2, S. 40) in der untersuchten Architektur zu finden oder die untersuchte Architektur mit Mustern zu vergleichen (siehe Kapitel 2, S. 8). Diese Einstiegshürde erfordert eine sehr gute und möglichst vollständige Pattern-Sammlung sowie die entsprechende Einarbeitungszeit oder Experten, welche die Inhalte der Patterns bereits kennen.

Die Interaktionsszenarios von SATURN werden aus Patterns extrahiert, die jeweils referenziert werden (Abschnitt 3.3, S. 31ff.). Somit erfolgt eine Vorauswahl von Patterns, die für eine Anforderung in Frage kommen. Allerdings (wie der nächste Abschnitt zeigt) ist die neue Methode nicht auf Patterns angewiesen.

Weitere Hilfsmittel wie Tabellenvorlagen sollen es ermöglichen, alle erhobenen Informationen zu dokumentieren. Wenn die Arbeitsschritte gut belegt sind, werden die Gütekriterien Nachvollziehbarkeit, Rückverfolgbarkeit und Wiederholbarkeit gewährleistet und die Auswertung vereinfacht. Dokumentationsvorschriften sollen deshalb in der neuen Methode konkret formuliert und eingehalten werden, z.B. indem laut Rollenmodell eine Person dafür verantwortlich ist.

4.1.6 *Forschungsfrage 6: Analyse offenlegende Fragetechnik mit geringeren Freiheitsgraden als frühere Arbeiten?*

Die früheren Methoden nutzen Fragetechniken (Abschnitte 2.4, S. 11ff. und 2.5, 15). Welche Fragen konkret bei der Analyse gestellt werden, kann aber nicht genauer erklärt werden, da diese von der jeweiligen Architektur und der jeweiligen Expertise der beteiligten Personen abhängen. Dieser hohe Freiheitsgrad soll verringert werden.

Sichtenmodelle verbinden verschiedene Perspektiven auf die Softwarearchitektur bezüglich Logik, Implementierung, Verteilung und Prozessen über Szenarien miteinander (siehe Abschnitt 4.2.1.4, S. 64 zur Beschreibung der Architektur).

Im Sinne der Sichtenmodelle soll die Tiefenanalyse eines Interaktionsszenarios architektonische Elemente und ihre Verbindungen zu Tage fördern, die verdeutlichen, ob und inwiefern ein Interaktionsszenario hypothetisch durchgeführt werden kann. Ein Vergleich mit Patterns, wie in den früheren Methoden, soll also nicht notwendig sein.

4.1.7 *Forschungsfrage 7, Teil 1: Wie voraussichtlichen Einfluss von Architekturentscheidungen auf Usability bestimmen und bewerten?*

VARIANTE 1 Es kann bestimmt werden, wie kritisch eine Entscheidung sein kann. Dazu ist eine Klassifizierung von Usability-Fehlern, also Fehlern bei der Benutzung, notwendig. Es kann differenziert werden zwischen den die Benutzung verhindernden Fehlern und den die Benutzung erschwerenden Fehlern [Koro7].

Vorteil: Die Unterscheidung zwischen die Benutzung verhindernder und erschwerender Fehler ermöglicht eine gute Priorisierung.

Nachteil: Es ist sehr schwierig, ohne Expertise zwischen beiden Fehlern zu entscheiden. Benutzungsfehler oder Szenarios müssen klassifiziert werden. Allerdings ist es abhängig vom Nutzungskontext (beispielsweise Hauptzweck und Alleinstellungsmerkmal), welche Szenarios für die Benutzer besonders wichtig sind. Die allgemeine Klassifikation kann deshalb irreführend sein.

VARIANTE 2 Die Beeinflussung von Usability-Attributen kann erörtert werden. Das bedeutet, dass für jede Entscheidung der theoretisch mögliche Einfluss auf ein definiertes Set von Attributen erfolgt (hier Effizienz, Effektivität, Erlernbarkeit, Erinnerungsfähigkeit, Sicherheit).

Vorteil: Die Begründung erfolgt detailliert und mit einem deutlichen theoretischen Bezug zur Usability. Wird eine Entscheidung so motiviert, kann die Begründung mit einer im Anschluss stattfindenden User-Evaluation überprüft werden.

Nachteil: Die Begründung zu einer einzelnen Entscheidung ist aufwendig und erfordert von Usability-Nichtexperten Wissen über Usability-Attribute, ihre Bedeutung und Interpretationsräume. Diese theoretische Argumentation käme zum eigentlichen Ziel, der Analyse der Architektur, hinzu. Gleichzeitig stellt sich aus praktischer Frage, wozu die Diskussion notwendig wäre. Aufwand und Nutzen stehen in keinem vertretbaren Verhältnis.

VARIANTE 3 Usability als Qualität ist darüber definiert, ob eine Software einer explizit geäußerten oder implizit existierenden Usability-Anforderung entspricht.

Vorteil: Dies ist aus theoretischer Sicht korrekt und einfach zu verstehen. Außerdem ist der Aufwand für die Teilnehmer von SATURN überschaubar: Der Einfluss der Entscheidungen auf die Usability kann über die einfach zu beantwortende Frage festgelegt werden, ob die Response eines Szenarios umgesetzt wird oder nicht. Es werden keine Vermutungen angestellt.

Nachteil: Diese Entscheidung ist zwar zielorientiert, zügig und eindeutig, aber dennoch wäre eine Abschätzung des Einflusses auf die Usability für eine Priorisierung von Szenarios sinnvoll.

Dieser Nachteil kann damit begegnet werden, dass bestimmt wird, ob die Response nicht oder nur teilweise umgesetzt wurde – so dass damit indirekt die Variante 1 abgebildet wird. Zur Priorisierung der Szenarios wird ergänzend der Kontext hinzugezogen (insbesondere Interaktionen, welche Hauptzweck und Alleinstellungsmerkmal betreffen). Das heißt, in der Methode werden am Ende der Phase 4 die Szenarios anhand der zusammengefassten Ergebnisse priorisiert.

Zudem kann nur eine User- oder Experten-Evaluation des User Interfaces dazu beitragen, die tatsächlichen Probleme und ihren Einfluss zu ermitteln.

AUSWAHL EINER VARIANTE Die dritte Variante ist am effizientesten, nachvollziehbar und korrekt. Die zweite Variante ist am aufwendigsten, bietet (zu) viel Raum für Interpretationen und ist nur unter wissenschaftlichen Gesichtspunkten interessant. Auch im Hinblick auf die Schwierigkeiten hinsichtlich der Isolierung der Einflüsse ist dieser Weg einer nachfolgenden wissenschaftlichen Forschungsarbeit zu überlassen. Die erste Variante kann erst nach einer allgemeinen und wissenschaftlich gesicherten Kategorisierung von Usability-Problemen erfolgen. Auch dies ist eine noch offene wissenschaftliche Frage, die Einflüsse verschiedener Nutzungskontexte zu berücksichtigen hat. Deshalb wird in SATURN die dritte Variante eingesetzt.

4.1.8 *Forschungsfrage 7, Teil 2: Wie potenziellen Änderungsaufwand der Softwarearchitektur bestimmen und bewerten?*

Eine Fragestellung von SATURN erfordert es, Änderungen an der Softwarearchitektur ihrem voraussichtlichen Aufwand nach zu klassifizieren. Dies ist notwendig, wenn aktuelle Architekturentscheidungen die Usability nicht unterstützen und wenn deshalb alternative Softwarearchitekturentscheidungen vorgeschlagen werden sollen.

Analog zur Bewertung der Unterstützung von Usability eignet sich also eine Benotung je nach Änderungsaufwand durch einen Architekten.

1. Typ 1 umfasst einfache Modifikationen des Verhaltens von einem oder mehreren architektonischen Elementen, die intern angepasst werden müssen.

   Nach der Definition von Softwarearchitektur von Kruchten [Kru98] sind diese Modifikationen wegen der Verhaltensänderung der architektonischen Elemente Architekturänderungen. Dazu gehören grundlegende Verhaltensänderungen in einer oder mehreren Komponenten, die je nach interner Struktur dieses architektonischen Elementes mehr oder weniger aufwendig sein können, und das Ändern von Schnittstellen und dessen Folgen.

2. Typ 2 umfasst komplexe Modifikationen, die durch das Ergänzen oder Entfernen von architektonischen Elemente(n) entstehen.

   Wenn eine oder mehrere architektonische Elemente ergänzt oder entfernt werden müssen, müssen meist eine oder mehrere bestehende architektonische Elemente modifiziert werden; sowohl Struktur als auch Verhalten der architektonischen Elemente werden geändert (nach der Definition von Softwarearchitektur von Kruchten [Kru98] sind dies also auch Architekturänderungen). Diese Modifikationen sind theoretisch komplexer und damit aufwendiger als Änderungen vom Typ 1.

3. Typ 3 umfasst unbekannte bzw. unvorhersehbare Änderungen, die von Typ 1 oder von Typ 2 sind.

   Um abzusichern, dass offene Fragen mit möglichem Einfluss auf die Softwarearchitektur mit in die Typisierung einfließen, wird der Typ 3 für unvorhersehbare Modifikationen bestimmt. Hier erfordern (unbekannte, aber notwendige) Modifikationen einen unbekannten Aufwand, der von Typ 1 oder 2 sein kann, nicht jedoch Typ 0 (dieser steht für „keine Modifikationen“).

4. Typ 0 bedeutet, dass keine Änderungen nötig erscheinen (Kategorie dient der Vollständigkeit).

Im Vergleich zu dieser Klassifikation werden in der Methode ATAM grob Nichtrisiken und Risiken unterschieden: Risiken sind Architekturentscheidungen, deren Einfluss

auf die Softwarearchitektur nicht bestimmt werden kann und die damit ein Risiko für die Softwarearchitektur darstellen [BCK03]. Dies entspricht dem Typ 3. In SALUTA gibt es keine Klassifikation von Architekturänderungen, sondern grundsätzlich nur die Aussage, dass eine Architektur die Usability einer bestimmten Anwendung unterstützt oder nicht unterstützt (Abschnitt 2.4, S. 11ff.). Die hier vorgestellte Klassifikation ist also, obwohl sie recht abstrakt gehalten ist, genauer als die Klassifikationen der früheren Arbeiten.

Die Klassifikation soll es vereinfachen, am Ende der Methode SATURN Prioritäten zu setzen und den Änderungsaufwand je nach betrachteter Architekturebene grob zu schätzen. Sie sollen Architekten, die sich mit den Analyseergebnissen konstruktiv auseinandersetzen, Anhaltspunkte für die kommende Aufgabe geben. Deshalb werden hier keine genaueren Bewertungen notwendig.

4.1.9 *Forschungsfrage 8: Wie iterative Erstellung der Architektur unterstützen?*

Der Entwurf der Architektur ist ein iterativer und zyklischer Prozess [Som01, Nus01, PBG04]. Das bedeutet, dass Analysen der Architektur ebenso zyklisch durchgeführt werden. Es ist also sinnvoll, die neue Methode so zu konstruieren, dass sie mit verschiedenen Versionen einer Architektur wiederholt wird. Da die Usability-Anforderungen an die Architektur gleich bleiben, sollen zumindest die gleichen Interaktionsszenarios untersucht werden.

Es ist also möglich, das bereits beschriebene Usability-Support-Level weiterzuführen. Dies ist auch sinnvoll, denn dann können abschließend die Interaktionsszenarios festgelegt werden, die sofort oder später in nachfolgenden Architekturversionen berücksichtigt werden sollen. Damit wird die Aussagekraft der Ergebnisse noch verstärkt, denn die Interaktionsszenarios werden zusätzlich priorisiert.

Wenn diese Priorisierung erst im letzten Arbeitsschritt erfolgt, kann es sein, dass notwendige Detailinformationen nicht mehr so stark bewusst sind. Deshalb soll schon während der Evaluation der Interaktionsszenarios bestimmt werden, in welcher folgenden Architekturversion ein Interaktionsszenario berücksichtigt sein soll.

Des Weiteren sind Austauschbeziehungen (oder „Wechselwirkungen“) zwischen verschiedenen Qualitätsmerkmalen beim Entwurf einer Softwarearchitektur zu beachten und zu balancieren [BCK03, CKK02]. Hier spielen auch Austauschbeziehungen von Usability mit anderen Qualitätsmerkmalen eine Rolle (siehe Einleitung in Kapitel 1, Abschnitt 1.2, Seite 2): genannt sind Sicherheit, Modifizierbarkeit und die Responsivität einer Anwendung auf Benutzereingaben.

Wie in Kapitel 2 (S. 8 ff.) beschrieben, werden andere Qualitätsmerkmale mit anderen Architekturanalysemethoden untersucht. Insbesondere von ATAM wird das Ziel verfolgt, Wechselwirkungen zwischen Qualitätsmerkmalen offenzulegen.

Um Architekten zu zeigen, welche weiteren Untersuchungen notwendig sein können, soll auch SATURN Austauschbeziehungen offenlegen. Es sollte daher anfangs in der Architekturbeschreibung aufgenommen werden, welche anderen Qualitätsmerkmale eine Rolle spielen (siehe Abschnitt 4.2.1.4 zur Architekturbeschreibung, S. 64); bei der Analyse sollen gegebenenfalls auffallende Wechselwirkungen mit anderen Qualitätsmerkmalen in den Einzeluntersuchungen und bei der Interpretation der Ergebnisse aufgeführt werden.

Qualitätsmerkmale werden in Qualitätsmodellen aufgeführt, deren Ziel es ist, Softwarequalität für Anwendungen zu spezifizieren und damit leichter überprüfbar zu machen [Int00]. Qualitätsmodelle unterteilen Qualität in Faktoren, die wiederum weitere Begriffe umfassen. Abhängigkeiten zwischen Faktoren werden nicht betrachtet. Die Perspektiven, aus denen die Qualitätsmodelle Softwarequalität betrachten, die Bezeichnung der Faktoren, die Anzahl der spezifizierten Begriffe sind unterschiedlich.

McCall et al. [MRW77] beschreiben aus Unternehmenssicht ein Modell mit elf Faktoren, die durch folgende Qualitätskriterien näher bestimmt werden.

- Korrektheit: Nachvollziehbarkeit, Vollständigkeit, Konsistenz
- Verlässlichkeit: Konsistenz, Akkuratheit, Fehlertoleranz
- Effizienz: Effizienz bei der Ausführung, Effizienz in der Datenhaltung
- Integrität: Zugriffskontrolle, Zugriffsprüfung
- Usability: Durchführbarkeit, Training, Kommunikativität
- Wartbarkeit: Einfachheit, Exaktheit, Selbstbeschreibungsfähigkeit, Modularität
- Testbarkeit: Einfachheit, Instrumentierung, Selbstbeschreibungsfähigkeit, Modularität
- Flexibilität: Selbstbeschreibungsfähigkeit, Erweiterbarkeit, Allgemeingültigkeit
- Portierbarkeit: Selbstbeschreibungsfähigkeit, Unabhängigkeit von einem Software-System, Maschinenunabhängigkeit
- Wiederverwendbarkeit: Selbstbeschreibungsfähigkeit, Allgemeingültigkeit, Modularität, Unabhängigkeit von einem Softwaresystem, Maschinenunabhängigkeit
- Interoperabilität: Modularität, Kommunikationsstandards (communication commonality), standardisierte Datenformate (data commonality)

Boehm et al. [BBK+81] beschreiben aus Entwicklersicht sieben Faktoren, die in drei Kategorien unterteilt sind: Aktuelle Nutzbarkeit (Reliabilität, Effizienz, Human Engineering), Wartbarkeit (Testbarkeit, Verständlichkeit, Modifizierbarkeit) und Portierbarkeit (eine Kategorie in sich).

- Nutzbarkeit
  - Verlässlichkeit: Abgeschlossenheit, Akkuratheit, Vollständigkeit, Robustheit/Integrität, Konsistenz
  - Effizienz: Verantwortlichkeit (accountability), Geräteeffizienz, Zugänglichkeit
  - Human Engineering (Usability): Robustheit/Integrität, Zugänglichkeit, Kommunikativität
- Wartbarkeit
  - Testbarkeit: Verantwortlichkeit (accountability), Kommunikativität, Selbstbeschreibungsfähigkeit, Strukturiertheit
  - Verständlichkeit: Konsistenz, Strukturiertheit, Exaktheit, Lesbarkeit
  - Modifizierbarkeit: Strukturiertheit, Verstärkbarkeit (augmentability)
- Portierbarkeit: Geräteunabhängigkeit

Der Standard ISO 9126 [Int00] beschreibt aus Benutzersicht sechs Faktoren zur Evaluation von Software, die in „Subfaktoren“ unterteilt sind:

- Portierbarkeit: Adaptierbarkeit, Installierbarkeit, Koexistenz, Ersetzbarkeit, Regelkonformität
- Verlässlichkeit: Reife, Fehlertoleranz, Wiederherstellbarkeit, Regelkonformität (compliance)

- Effizienz: Zeitverhalten, Ressourcenverhalten, Regelkonformität (compliance)
- Wartbarkeit: Analysierbarkeit, Modifizierbarkeit, Stabilität, Testbarkeit, Regelkonformität
- Funktionalität: Angemessenheit, Sicherheit, Akkuratheit, Interoperabilität
- Usability: Verständlichkeit, Erlernbarkeit, Nützlichkeit (operability), Attraktivität, Regelkonformität

Die Modelle sind sich inhaltlich sehr ähnlich, obwohl sie aus verschiedenen Perspektiven erstellt wurden. Alle drei klassifizieren Usability als Softwarequalitätsmerkmal. Tabelle 19 stellt diese drei Qualitätsmodelle gegenüber, indem inhaltlich ähnliche Faktoren in einer Zeile zusammengefasst werden. Daraus entsteht eine zusammenfassende Liste von 14 Qualitätsmerkmalen.

Dabei ist zu beachten, dass manche Faktoren in den Qualitätsmodellen zwei oder mehr Qualitätsmerkmale beeinflussen: beispielsweise ist Regelkonformität im Modell der ISO [Into0] sowohl ein Faktor für Portierbarkeit als auch für Verlässlichkeit, Effizienz und Usability.

Qualitätsmerkmal	*McCall et al.* (Unternehmenssicht)	*Boehm et al.* (Entwicklersicht)	*ISO 9126* (Benutzersicht)
Korrektheit	Korrektheit	Korrektheit	Wartbarkeit
Funktionalität	–	–	Funktionalität
Verlässlichkeit	Verlässlichkeit	Verlässlichkeit	Verlässlichkeit
Effizienz	Effizienz	Effizienz	Effizienz
Integrität	Integrität	Integrität	–
Usability	Usability	Usability	Usability
Wartbarkeit	Wartbarkeit	–	Wartbarkeit
Modifizierbarkeit	–	Modifizierbarkeit	Wartbarkeit
Testbarkeit	Testbarkeit	Testbarkeit	Wartbarkeit
Flexibilität	Flexibilität	Modifizierbarkeit	Flexibilität
Portierbarkeit	Portierbarkeit	Portierbarkeit	Portierbarkeit
Wiederverwendbarkeit	Wiederverwendbarkeit	Portierbarkeit	Portierbarkeit
Interoperabilität	Interoperabilität	–	–
Verständlichkeit	–	Verständlichkeit	–

Tabelle 19: Qualitätsmodelle im Überblick (inhaltlich ähnliche Faktoren in einer Zeile)

4.1.10 *Forschungsfrage 9: Wie interdisziplinäre Kooperation unterstützen?*

Mit einer Softwarearchitekturanalysemethode kann die architektonische Unterstützung von Usability, also der Informationsarchitektur, dem User-Interface und dem Interaktionsdesign einer Anwendung untersucht werden. Die Qualität der Benutzung selbst wird schließlich mit Nutzer-Evaluationen untersucht, wobei die Probleme bei der Nutzung deutlich werden. Usability-Attribute bilden hier die Indikatoren für Usability-Probleme. Ergebnisse von Nutzertests weisen auf Fehler im Design hin, sei es die Informationsarchitektur, das User-Interface, das Interaktionsdesign, Hardware oder Software-Fehler; sie müssen interpretiert werden, um Ursachen für Usability-Probleme zu erkennen.

Um eine Softwarearchitekturanalysemethode und einen Nutzertest miteinander kombinieren zu können, sollen die Evaluationsparameter auf den gleichen Usability-Anforderungen basieren. Als gemeinsame Grundlage eignet sich dafür der Nutzungs-

kontext, denn von diesem werden Testaufgaben für die Nutzer abgeleitet. Mit diesem Annähern an Arbeitsweisen und Sichtweisen soll der Informationsfluss zwischen Softwaretechnik und Usability Engineering unterstützt werden, der sonst maßgeblich behindert werden würde [FJM05].

4.2 BESCHREIBUNG DER METHODE SATURN

Mit der szenario-basierten Methode „Softwarearchitekturanalyse von Usability-Anforderungen“[3] SATURN wird analysiert, wie Usability architektonisch unterstützt wird. Ermittelt werden Usability-beeinflussende Architekturentscheidungen und das entsprechende Usability-Support-Level der untersuchten Softwarearchitektur.

4.2.1 *Metamodell*

Das Metamodell besteht aus dem Qualitätsmodell von SATURN und dem Argumentationspfad.

4.2.1.1 *Qualitätsmodell von SATURN*

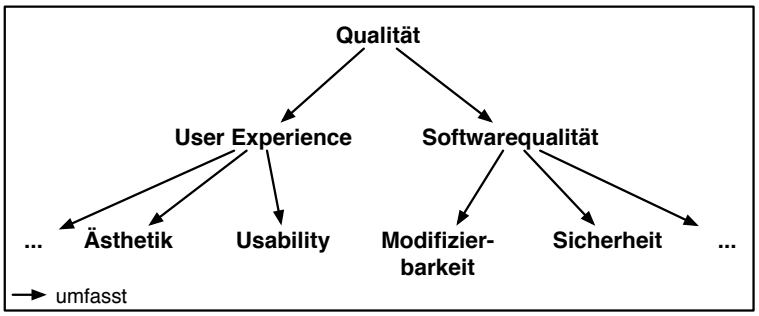

Abbildung 13: Qualitätsmodell von SATURN

Softwarequalität ist (1) das Ausmaß, zu dem ein System, eine Komponente oder ein Prozess spezifizierten Anforderungen entspricht, (2) das Ausmaß, zu dem ein System, eine Komponente oder ein Prozess den Erwartungen und Bedürfnissen von Kunden oder Nutzern entspricht: „Software quality is (1) the degree to which a system, component, or process meets specified requirements. (2) The degree to which a system, component, or process meets customer or user needs or expectations.“ [IEE90]

In Softwarequalitätsmodellen wird Usability als Softwarequalität klassifiziert [MRW77, BBK+81, Int00] (siehe Anhang A.3, S. 255). Usability (Benutzbarkeit) ist allerdings, genau genommen, die von Menschen erlebte Qualität der Benutzung [Bev01] und kann damit nicht als Softwarequalität bezeichnet werden. Usability definiert, zu welchem Ausmaß bestimmte Personen zur Erreichung bestimmter Ziele in einem bestimmten Nutzungskontext ein Produkt verwenden können [Int99]. Usability ist wie Ästhetik und Spaß ein Teilziel zur Erreichung eines guten Nutzungserlebnisses (User Experience) [PRS07].

In diesem Sinne ist die Unterstützung für Usability ein Softwarequalitätsmerkmal und nicht die Usability an sich. Besonders wichtig ist in diesem Zusammenhang, dass zwischen verschiedenen Qualitätsmerkmalen beim Entwurf einer Softwarearchitektur Austauschbeziehungen zu beachten und zu balancieren sind (Abschnitt 1.2 ab S. 2, Abschnitt 4.1.9 ab S. 58 und dort die Tabelle 19, auf Seite 61).

3 Engl. “Software ArchitecTure analysis of Usability-RequiremeNts” SATURN

4.2.1.2 *Spezifikation von Usability-Anforderungen in SATURN*

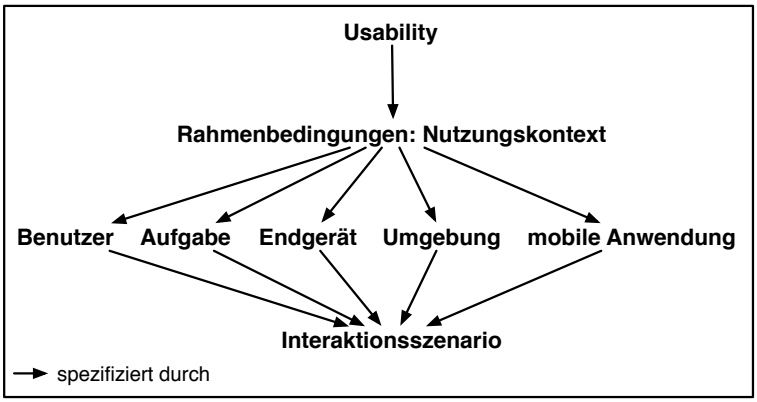

Abbildung 14: Usability spezifizieren in SATURN

Im Usability Engineering folgen zyklisch und iterativ Anforderungserhebung, Design und Evaluation aufeinander ab [Nie93]. In der Methode SATURN soll untersucht werden, inwiefern Usability-Anforderungen umgesetzt werden. Anforderungen werden generell unterteilt in funktionale Anforderungen, Rahmenbedingungen und Qualitätsanforderungen [Poh06].

Der Begriff Usability steht für Benutzbarkeit und beschreibt die von Menschen erlebte Qualität der Benutzung [Bev01]. Usability definiert, zu welchem Ausmaß bestimmte Personen zur Erreichung bestimmter Ziele in einem bestimmten Nutzungskontext ein Softwaresystem verwenden können [Int99]. Usability ist neben Zielen wie Ästhetik und Spaß ein Teilziel zur Erreichung eines guten Nutzungserlebnisses (User Experience) [PRS07].

Die Qualitätsanforderung Usability wird in SATURN verfeinert, in dem die Rahmenbedingungen ermittelt werden, von denen dann funktionale Anforderungen, weitere Rahmenbedingungen und Qualitätsanforderungen abgeleitet werden können.

Der Nutzungskontext umfasst Benutzer, die in verschiedenen Umgebungen agieren und mit dem Endgerät bestimmte Aufgaben erledigen, für die sie teilweise oder vollständig die mobile Anwendung nutzen. Was sich hinter einzelnen Kontextfaktoren (siehe Abschnitt 4.1.2, S. 48ff.) verbirgt, wurde mittels einer Literaturstudie und Fallstudien ermittelt.

Ein Interaktionsszenario (Kapitel 3, S. 27ff.) beschreibt Interaktionen [DFAB03] zwischen Nutzer und System und enthält eine funktionale Anforderung aus Sicht einer *Quelle*: eine bestimmte benutzer- oder systeminitiierte Interaktion, die mit der mobilen Anwendung durchgeführt werden soll. Diese wird als *Stimulus* an ein System (*Artefakt*) in einer bestimmten *Umgebung* gesendet. Diese Umgebung bzw. Umgebungen haben verschiedene technische Eigenschaften und soziale Rahmenbedingungen. Die Reaktion des Artefakts auf den Stimulus, die *Response*, stellt eine weitere funktionale Anforderung dar, denn sie beschreibt, was geschehen soll. Qualitätsanforderungen, die durch die Rahmenbedingungen des Nutzungskontexts bestimmt werden, sind im Abschnitt *Response-Prüfung* aufgeführt. Das Format der Interaktionsszenarios bezieht sich auf ein schon etabliertes Szenario-Format [BJK01], das aus dem Qualitätsattributs-Kontext in den Anforderungskontext überführt wurde.

4.2.1.3 *Argumentationspfad*

Abbildung 15: Argumentationspfad in SATURN

Mit Use Cases [Obj11, Sto05] werden konkrete Funktionaliäten von definierten Rollen eines Systems in einem bestimmten Format beschrieben [JBR99]. Als Typ von Szenarios eignen sich Use Cases, die abstrakte Interaktionsszenarios in detailliertere technische Anforderungen überführen. Ziel ist es, durch eine Architekturanalyse im Sinne der Sichtenmodelle konkrete Architekturentscheidungen textuell zu formulieren und ggf. (mit der UML) zu illustrieren.

„Eine Architektur ist die Menge von signifikanten Entscheidungen über die Organisation eines Softwaresystems, die Auswahl der strukturellen Elemente und ihrer Schnittstellen, durch die das System zusammengesetzt ist, ihr Verhalten, welches spezifiziert wird durch die Kollaborationen zwischen diesen Elementen, die Zusammensetzung dieser strukturellen und verhaltensbeeinflussenden Elemente in progressiv größer werdende Subsysteme und der Architekturstil, der diese Organisation leitet." [Kru98]

Architekturentscheidungen bestimmen die Struktur und das Verhalten des Systems und setzen einen für das finale Softwaresystem bindenden Rahmen [Kru04]. Sie beeinflussen die Softwarequalität [BCK03], indem sie die Durchführung eines Interaktionsszenarios fördern oder behindern.

Architekturentscheidungen, die ein konkretes Interaktionsszenario positiv oder negativ beeinflussen, werden als szenario-beeinflussende Architekturentscheidungen (S) bezeichnet. Diese szenario-beeinflussenden Architekturentscheidungen sind entweder positiv oder negativ. Letztere können zu architektursensitiven Architekturentscheidungen führen, die eine negative szenario-beeinflussende Architekturentscheidung ersetzen sollen; um solche Fälle zu erkennen, werden solche Alternativen entsprechend ihres voraussichtlichen Änderungsaufwandes gekennzeichnet.

4.2.1.4 *Beschreibung der Architektur*

Nach [BCK03, HS09] enthält eine Architekturbeschreibung neben den technischen Details auch Informationen zu wirtschaftlichen und technischen Rahmenbedingungen:

- Wirtschaftliche Rahmenbedingungen: Geschäftsumfeld, Stakeholder, Ziel und Hauptzweck der Anwendung, Ressourcen (Budget, Zeit, Personal), organisatorische und juristische Standards, Konventionen, Softwareentwicklungsprozess (beispielsweise V-Modell, Extreme Programming)
- Technische Rahmenbedingungen: Interoperation mit anderen Systemen (technische Einbettung), Hardware-Vorgaben, Software-Vorgaben (beispielsweise COTS-Komponenten, Legacy-Code), Vorgaben Systembetrieb, Programmiervorgaben
- Qualitätseigenschaften: Liste der relevanten Qualitätsmerkmale und von welchen Geschäftszielen diese abgeleitet sind (Tabelle 19, S. 61)
- Ansprechpartner

Die Softwarearchitektur einer Anwendung komplett zu beschreiben, ist oftmals eine komplexe und sehr aufwendige Aufgabe [Kru98, CBB+02]; üblich ist es daher, die

Architektur in verschiedenen Abstraktionsebenen und aus verschiedenen Perspektiven textlich und/oder grafisch darzustellen.[4]

Wie detailliert eine Softwarearchitektur beschrieben werden muss, hängt ab von der Größe des Systems, von den betrachteten Qualitätsmerkmalen und dem Maß an Gewissheit und Kontrolle, das erreicht werden soll [Bos00, PBG04]. Der Abstraktionsgrad der Gesamtarchitektur des Softwaresystems einer Raumsonde ist im Vergleich zu dem einer mobilen Anwendung größer. Werden Qualitätsmerkmale wie Laufzeitverhalten, Sicherheit und Usability betrachtet, so erfordern diese verschiedene Perspektiven auf das Softwaresystem und entsprechende Abstraktionsgrade. Da schlechtes Quellcode-Design und schlechte Implementierung verhindern können, dass Anforderungen umgesetzt werden, kann bis zur Quellcode-Ebene geprüft werden, um ein hohes Maß an Gewissheit und Kontrolle über die Qualität zu erreichen [PBG04].

Als Architekturbeschreibungsebenen beschreibt Starke [Sta05] mehrere Bausteinschichten vom Allgemeinen zum Detaillierten. Inhalte einzelner Black-Box-Bausteine eines Diagramms werden dabei jeweils in einem genauer beschriebenen Diagramm bestimmt. Die letzte Abstraktionsstufe ist dabei der Quellcode.

Gruhn und Thiel [GT00] unterscheiden drei Abstraktionsgrade: die fachliche, die softwaretechnische und die systemtechnische Architektur. In der fachlichen Architektur werden Geschäftsobjekte und Geschäftskomponenten betrachtet, also logisch zusammenhängende Teile der zukünftigen Anwendung wie Datenmodelle, Klassen und deren Beziehungen. Die softwaretechnische Architektur beschreibt Softwarekomponenten, die Funktionen der fachlichen Architektur umsetzen; der Komponentenschnitt fasst dabei fachliche Objekte zusammen, die auf technischer, lauffähiger Software abgebildet werden. Eine fachliche Komponente kann von mehreren Softwarekomponenten implementiert werden. Die systemtechnische Softwarearchitektur beschreibt die Implementierungsdetails, beispielsweise die Ablaufumgebung, das Betriebssystem, die Datenbank und den Applikationsserver.

Im 4+1 Modell der Softwarearchitektur [Kru98] und ähnlichen Modellen [CKK02] werden Perspektiven auf die Softwarearchitektur bezüglich Logik, Implementierung, Verteilung und Prozessen über Szenarios miteinander verknüpft. Szenarios beschreiben ein Modell möglicher Ereignisse [WKRSS09], also die hypothetische Durchführung einer Sache.[5]

So unterscheidet das 4+1 Modell [Kru98] die Logische Sicht (Objektmodell), die Prozesssicht (Konkurrenz und Synchronisation), die Physikalische Sicht (Abbildung der Software auf Hardware), die Entwicklungssicht (statische Sicht und Einbindung in die Entwicklungsumgebung) und die Szenarios. Szenarios beschreiben Anforderungen, deren Umsetzung mit Hilfe der Sichten genauer beschrieben werden kann.

Das Modell von Hofmeister, Nord und Soni [HNS99] unterscheidet die konzeptionelle Sicht (Komponenten und Konnektoren), die Modulsicht (Subsysteme, Module) und die Code-Sicht (Dateien, Ordnerstruktur, Bibliotheken). Diese Sichten sind mit der Ausführungssicht verbunden (wie beispielsweise Threads) und werden durch den Quellcode repräsentiert. Ressourcen der Ausführungssicht werden durch die Hardware-Sicht (Prozesse, Netzwerk, Disks) dargestellt. Diese Sichten-Modelle sind sich ähnlich, wobei das zweite mehr Abstraktionsgrade umfasst.

- Architekturebene 0: fachliche Datenmodelle (logische Sicht) und Szenarios; hohes Abstraktionsniveau, denn Nutzfälle und Rahmenbedingungen werden betrachtet; ein fachliches Datenmodell kann später in ein Datenmodell des Softwaresystems überführt werden

- Architekturebene 1: Komponenten und Schnittstellen der obersten Ebene

4 Eine Vorlage zur Beschreibung der Softwarearchitektur ist verfügbar unter http://www.arc42.de/ [HS09].
5 Duden, 18.4.2012, `http://www.duden.de/rechtschreibung/Szenario`

- Architekturebene 2: Komponenten und Schnittstellen der nächsten Ebene(n) (Black-Box-Bausteine eines Diagramms werden in einem genauer beschriebenen Diagramm bestimmt)
- Architekturebene Feindesign: genaue Beschreibung einzelner Komponenten und Schnittstellen, aber noch nicht Quellcode
- Architekturebene Quellcode: ausführbare Programmanweisungen

In den Architekturebenen 1, 2 und Feindesign können strukturelle Fragen mit der logischen und physikalischen Sicht (Kruchten) bzw. der konzeptionellen Sicht und der Modulsicht (Hofmeister, Soni, Nord) betrachtet werden; das Verhalten bildet die Prozesssicht (Kruchten) bzw. die Ausführungssicht (Hofmeister, Soni, Nord) ab. Die Ebene Implementierung wird durch den Quellcode repräsentiert.

In SATURN werden Architekturentscheidungen meist textlich beschrieben. Wenn ein System teilweise oder komplett entwickelt ist, ist es sinnvoll, die Struktur des tatsächlich implementierten Systems zu visualisieren. Damit wird der tatsächliche Status der Architektur sichtbar, der vom konzipierten Status abweichen kann. Dazu eignen sich zum Beispiel generierte Diagramme, wie Paketdiagramme und Klassendiagramme.[6] Zusätzlich können deshalb Architekturbeschreibungssprachen („Sprachen“) verwendet werden [MT00].

Dazu gehören grafische Modellierungssprachen wie beispielsweise die Unified Modeling Language (UML) [Obj11], die Sichten und Szenarios über bestimmte Diagrammtypen visualisieren[7]. Die UML ist eine grafische Modellierungssprache, mit der Softwaresysteme, Geschäftsprozesse und Datenstrukturen dokumentiert, spezifiziert und konstruiert werden können. Ein Modell ist eine Beschreibung oder eine Spezifikation des Systems und dessen Umgebung zu einem bestimmen Zweck, das durch Diagramme sowie natürlichsprachlichen oder formalen Text repräsentiert wird. Die UML umfasst Struktur- und Verhaltensdiagramme, die mit natürlichsprachlichen oder formalen Notizen versehen werden können. Strukturdiagramme zeigen die statische Struktur des modellierten Systems (Klassen-, Komponenten-, Verteilungs- und Paketdiagramm). Verhaltensdiagramme beschreiben das dynamische Verhalten zwischen Elementen eines Systems, also ihre Kollaborationen und Aktivitäten (Aktivitäts-, Anwendungsfall-, Sequenz-, Zeitverlaufs- und Zustandsdiagramm). Szenarios sind das verbindende Element: mit Diagrammen der anderen vier Sichten werden Struktur und Verhalten der architektonischen Elemente identifiziert und beschrieben.

- Logische Sicht: zeigt insbesondere, wie die Architektur funktionale Anforderungen der Nutzer umsetzt; UML-Diagramme: z. B. Klassen-, Kommunikations- und Sequenzdiagramm
- Implementierungssicht: zeigt den Aufbau des Systems für Entwickler; UML-Diagramme: z. B. Komponenten-, Paketdiagramm
- Prozesssicht: zeigt das Laufzeitverhalten des Systems, zum Beispiel Prozesse und deren Kommunikation; UML-Diagramm: Aktivitätsdiagramm
- Verteilungssicht: zeigt die Verteilung der Elemente auf der Hardware und die Kommunikation zwischen ihnen; UML-Diagramm: Verteilungsdiagramm
- Szenarios: zeigen Anwendungsfälle auf; UML-Diagramm: Anwendungsfalldiagramm

6 Beispielsweise können diese Dokumentationen durch annotierten Quelltext mit der quelloffenen Software „Doxygen“ erstellt werden `http://www.stack.nl/~dimitri/doxygen/`.

7 Ein Standardwerk zur UML 2 ist beispielsweise [Sto05]. Eine vierseitige Kurzreferenz zur UML 2 ist erhältlich via `http://www.oose.de/downloads/umlnotationsuebersicht2.pdf`.

4.2.2 *Kurzübersicht von SATURN*

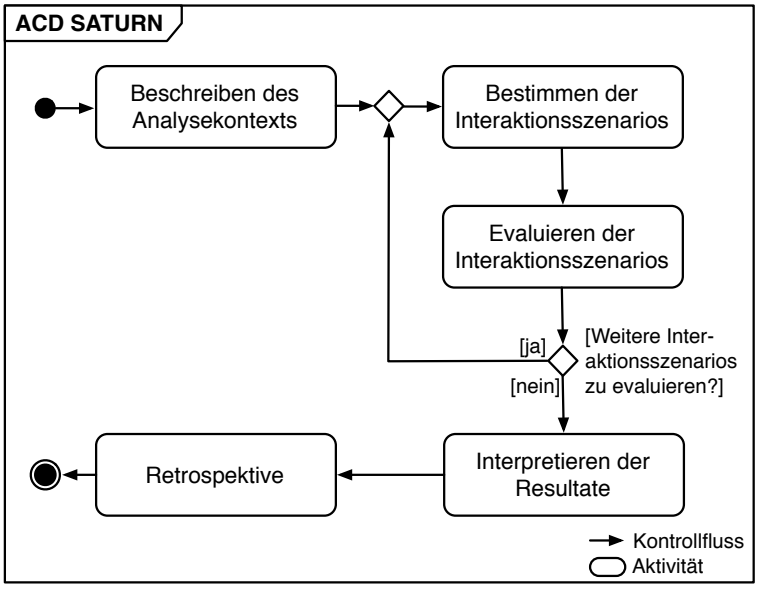

Abbildung 16: Aktivitätsdiagramm zu SATURN

Hauptaktivitäten von SATURN sind die Phasen (1) Beschreiben des Analysekontexts, (2) Bestimmen der Interaktionsszenarios, (3) Evaluieren der Interaktionsszenarios, (4) Interpretieren der Resultate und (5) Retrospektive (siehe Abbildung 16).

(1) Beschreiben des Analysekontexts (Abschnitt 4.2.3, S. 68ff.): Grob wirtschaftlichen Kontext und Nutzungskontext bestimmen; Interaktionen vom Nutzungskontext ableiten;

(2) Bestimmen der Interaktionsszenarios (Abschnitt 4.2.4, S. 68ff.): Interaktionsszenarios aus dem bereitgestellten Katalog auswählen; zu evaluierende Interaktionsszenarios bestimmen;

(3) Evaluieren der Interaktionsszenarios (Abschnitt 4.2.5, S. 69ff.): für jedes Interaktionsszenario getroffene Architekturentscheidungen ermitteln und evaluieren, inwiefern sie zum jeweiligen Interaktionsszenario konform sind; (optional: falls sie das nicht sind, alternative Architekturentscheidungen brainstormen);

(4) Interpretieren der Resultate (Abschnitt 4.2.6, S. 71ff.): Ergebnisse der Phase 3 zusammenfassen, übergreifende Themen ermitteln und schematisch diskutieren

(5) Retrospektive (Abschnitt 4.2.7, S. 73ff.): über das Vorgehen und die Ergebnisse der Methode, mit dem Ziel, die Methode und die Interaktionsszenarios kontinuierlich zu verbessern.

Um den Ressourcenaufwand, der dazu benötigt wird, Interaktionsszenarios von Grund auf zu erstellen, zu verringern, sind 50 allgemeine Interaktionsszenarios vorbereitet (Anhang: Abschnitt A.2, S. 204ff.; sowie Kapitel 3, S. 27ff.).

Als Rollenmodell werden folgende Verantwortungsbereiche von Analysten und Architekten bestimmt:

- Architekten kennen die Anwendung, die Wünsche der Stakeholder, alle Dokumente, Informationen und Ansprechpartner. Sie führen daher die Tiefenanalyse durch, beantworten Fragen der Analysten.

- Analysten kennen die Interaktionsszenarios, bereiten die Dokumentation für jede Phase vor und achten darauf, dass der Fokus auf die Analyse gewahrt bleibt. Sie fassen abschließend die Erkenntnisse über die Methode zusammen, aktualisieren die Interaktionsszenarios und/oder die Methode.
- Analysten und Architekten formulieren gemeinsam die Inhalte, die in die Evaluationsunterlagen aufgenommen werden (z. B. Nutzungskontext, evaluierte Interaktionsszenarios, Architekturentscheidungen, Zusammenfassungen).

4.2.3 *SATURN-Phase 1: Beschreiben des Analysekontexts*

VORBEDINGUNG Die Softwarearchitektur soll ausreichend dokumentiert sein, damit die Methode SATURN durchgeführt werden kann. Folgende Kriterien sollen erfüllt sein.

- Wirtschaftliche und technische Rahmenbedingungen sollen bekannt sein (siehe Abschnitt 4.2.1.4 auf S. 64).
- Je mehr Informationen über Struktur und Verhalten der architektonischen Elemente vorliegen, umso genauer kann die Anwendung untersucht werden. Ein Datenmodell und wie es in den architektonischen Elementen repräsentiert sein soll, reicht nicht aus: aus der Architektur soll hervorgehen, wie Nutzer-System-Interaktionen umgesetzt werden sollen. Dazu sollen die Architekturebenen 1 und 2 skizziert und/oder beschrieben sein (siehe Abschnitt 4.2.1.4 auf S. 65).
- Falls vorhanden, sollen der Nutzungskontext, Personas, Skizzen zu Interaktionen und zum User Interface, die Informationsarchitektur, Prototypen und ggf. Ergebnisse von Nutzer-Evaluationen oder Kundenbefragungen in der ersten Phase hinzugezogen werden.

ANALYSEKONTEXT BESCHREIBEN Der Analysekontext umfasst eine kurze Beschreibung des wirtschaftlichen Kontexts und des Nutzungskontexts.

- Wirtschaftlicher Kontext: Hauptverwendungszweck und Alleinstellungsmerkmal enthalten die Funktionen und Interaktionen, die von Nutzern explizit oder implizit erwartet werden.
- Nutzungskontext: Als Parameter des Nutzungskontexts werden die Aufgaben der Nutzer, Umgebungstypen, Benutzertypen und Endgerätetypen der mobilen Anwendung grob definiert.

WALKTHROUGH UND BRAINSTORMING In einem Walkthrough durch die aktuelle Benutzerschnittstelle werden ein oder mehrere Anwendungsfälle von Anfang bis Ende durchgespielt, um Nutzer-System-Interaktionen dieser Anwendung aufzulisten. Diese Liste wird durch ein Brainstorming erweitert, bei dem jede Person stichwortartig Interaktionen der Anwendung auflistet.

4.2.4 *SATURN-Phase 2: Bestimmen der Interaktionsszenarios*

VORAUSWAHL VON INTERAKTIONSSZENARIOS Jeder Teilnehmer der Analyse wählt aus den 50 Interaktionsszenarios ein erstes Set aus und begründet seine Auswahl kurz. Die Interaktionsszenarios sollen Interaktionen beschreiben, mit denen der Nutzer seine Aufgaben mit der Anwendung durchführen kann und die wahrscheinlich architektonische Unterstützung in dieser Anwendung benötigen.

Aus Nutzersicht muss der Hauptzweck der Anwendung erfüllt werden, aus wirtschaftlicher Sicht ist das Alleinstellungsmerkmal gegenüber der Konkurrenz wesentlich, weshalb die entsprechenden Interaktionsszenarios obligatorisch sind. Der Analyst wählt mindestens diese Interaktionsszenarios aus.

NEUE INTERAKTIONSSZENARIOS Neue Interaktionsszenarios (Definition siehe Abschnitt 3.3.1, S. 31) können kurz formuliert und später mit der Vorlage in Tabelle 61 (Anhang A.1, S. 203) verfeinert und mit dem Lebenszyklus in Abbildung 11 (S. 45) gepflegt werden, um ein einheitliches Abstraktionsniveau der Interaktionsszenarios zu erreichen.

ENDGÜLTIGE AUSWAHL DER INTERAKTIONSSZENARIOS Auswahlgründe für Interaktionsszenarios werden diskutiert und dokumentiert, um nachvollziehbar zu einem Set für die Analyse zu gelangen.

ERSTELLUNG PROJEKTSPEZIFISCHER INTERAKTIONSSZENARIOS

4.2.5 *SATURN-Phase 3: Evaluieren der Interaktionsszenarios*

ARCHITEKTURENTSCHEIDUNGEN ERMITTELN In dieser Methode werden – analog zum Verständnis von Sichtenmodellen (Abschnitt 4.2.1.4, S. 64ff.) – zu Anwendungsfällen der Interaktionsszenarios architektonische Elemente und ihr Verhalten ermittelt und bewertet. Für den Ablauf eines Interaktionsszenarios werden Teil-Anwendungsfälle stichwortartig aufgelistet. Anschließend werden die architektonischen Elemente und Schnittstellen ermittelt, welche für die Teil-Anwendungsfälle verantwortlich sind. Zur Analyse werden immer die gleichen Fragen gestellt, deren Antworten im Evaluationsformular tabellarisch kurz aufgelistet werden:

1. Welche architektonischen Elemente interagieren miteinander, um den Anwendungsfall durchzuführen? (z. B. Komponenten, Schnittstellen)
2. Wie interagieren sie miteinander? (z. B. über Protokolle, in einer bestimmten Sequenz, über bestimmte Daten- und Kontrollflüsse)
3. Sind sie maßgeblich für die Response des Interaktionsszenarios verantwortlich? Wenn ja, dann werden entsprechende Architekturentscheidungen ausformuliert: Diese szenario-beeinflussenden Architekturenscheidungen (S) sind entweder positiv (in dem Fall sind keine Änderungen nötig) oder negativ (ggf. architektursensitive Änderungen notwendig).
4. Können die Architekturentscheidungen auf andere Qualitätsmerkmale positiv oder negativ wirken? Diese Frage betrifft die anderen Qualitätsmerkmale der Architektur[8].

ARCHITEKTURENTSCHEIDUNGEN PRÜFEN Die Architekturentscheidungen sollen die Response des Interaktionsszenarios nicht beeinträchtigen. Wenn ein Interaktionsszenario nicht durchgeführt werden und damit die Anwendung nicht mehr benutzt werden kann, ist das ein großer negativer Einfluss auf die Usability. Wird ein Interaktionsszenario nur teilweise behindert, ist die Anwendung noch benutzbar, aber nicht so gut, wie spezifiziert. Dies wird als kleiner negativer Einfluss auf die Usability gewertet.

Tabelle 20 enthält die Bewertung der architektonischen Beeinträchtigungen von Usability.

8 Falls keine anderen Qualitätsmerkmale in der Architekturbeschreibung aufgeführt wurden, können Qualitätsmodelle zu Rate gezogen werden (siehe Abschnitt A.3, S. 255ff.)

Typ	*Beschreibung*	*Einfluss auf Usability*
0	Unterstützt das Interaktionsszenario	Response und Response-Prüfung werden unterstützt; kein negativer architektonischer Einfluss auf die Usability ersichtlich
1	Unterstützt das Interaktionsszenario teilweise	Response und/oder Response-Prüfung werden teilweise unterstützt; kleinerer negativer architektonischer Einfluss auf Usability identifiziert
2	Unterstützt das Interaktionsszenario nicht	Response und Response-Prüfung werden nicht unterstützt; großer negativer Einfluss auf Usability identifiziert
3	Unterstützt das Interaktionsszenario teilweise oder nicht	unbekannt, inwiefern Response und Response-Prüfung nicht unterstützt werden; architektonischer Einfluss auf Usability kann Typ 1 oder 2 sein, nicht jedoch Typ 0

Tabelle 20: Einfluss einer Architekturentscheidung auf Usability

Ist eine Architekturentscheidung nicht konform, können konforme Architekturentscheidungen konstruktiv erarbeitet werden. Der Änderungsaufwand, den die neuen Architekturentscheidungen erfordern, wird grob nach seiner Komplexität bewertet.

Wenn architektonische Elemente hinzugefügt, entfernt oder anders kombiniert werden sollen, werden diese Änderungenen als komplexer eingestuft als Änderungen, die ein architektonisches Element betreffen (und damit tiefere Architekturebenen) oder Änderungen, die das Verhalten von architektonischen Elementen untereinander betreffen (wenn keine Strukturänderung nötig ist). Tabelle 21 fasst die Bewertungen des benötigten Änderungsaufwandes zusammen.

Typ	*Beschreibung*	*Änderungsaufwand*
0	Keine Modifikationen	keine Änderungen notwendig
1	Einfache Modifikationen	Verhalten einer oder mehrerer Komponenten und/oder Schnittstellen soll geändert werden; Modifikationen erfordern keine strukturellen Änderungen
2	Komplexe Modifikationen	Ein oder mehrere architektonische Elemente müssen hinzugefügt, entfernt oder anders kombiniert werden; Modifikationen betreffen sowohl Struktur als auch Verhalten der architektonischen Elemente
3	Unvorhersehbare Komplexität	Unbekannter Aufwand, kann von Typ 1 oder 2 sein, nicht jedoch von Typ 0

Tabelle 21: Komplexität von architektonischen Änderungen

Es obliegt den Architekten, die Änderungsvorschläge anzunehmen oder in konstruktiven Phasen bei der Erstellung der Softwarearchitektur andere Lösungen zu integrieren.

Die Architekturentscheidungen werden separat und fortlaufend aufgelistet, um ihre thematische Umsortierung in Phase 4 vorzubereiten.

INTERAKTIONSSZENARIO ABSCHLIESSEN Am Ende wird das Interaktionsszenario bewertet. Da Durchschnittswerte offengelegte, nicht konforme Architekturentscheidungen verschleiern würden, gelten die höchsten Bewertungen des Einflusses auf die Usability und des architektonischen Änderungsaufwandes für das untersuchte Interaktionsszenario. Abschließend werden Zielbewertungen aufgeführt, die bei einer Wiederholung der Architekturanalyse mit späteren Versionen der Softwarearchitektur erreicht werden sollen.

WEITERE DURCHGÄNGE Nachdem alle Interaktionsszenarios evaluiert wurden, kann es auch Gründe geben, noch einmal zur Phase 2 zurückzugehen.

- Abgesprochenes Vorgehen: Es wurden im Vorfeld nur wenige Interaktionsszenarios ausgewählt, weitere später.
- Lerneffekte: Es stellte sich heraus, dass die vorhandene Architektur erfordert, weitere Interaktionsszenarios zu betrachten.
- Ergebnisse verfeinern: Es wird gewünscht, die bisherigen Ergebnisse noch einmal zu prüfen, zu erweitern, zu überarbeiten.

4.2.6 *SATURN-Phase 4: Interpretieren der Resultate*

ARCHITEKTUR-SUPPORT-LEVEL Um nachzuvollziehen, wie die architektonische Unterstützung aller Interaktionsszenarios bewertet wurde, wird zudem die Architektur-Support-Level-Tabelle (ASL-Tabelle) aufgestellt.

Zum einen kann es durch Architekturänderungen dazu kommen, dass schon einmal evaluierte Interaktionsszenarios nicht mehr architektonisch unterstützt werden. Zum anderen sollen die Analyseergebnisse von verschiedenen Architekturversionen vergleichbar sein. Deshalb ist es sinnvoll, schon einmal verwendete Interaktionsszenarios bei der Analyse einer neuen oder einer alternativen Architekturversion einer Anwendung erneut zu prüfen. Aufgrund dessen listet die ASL-Tabelle die Bewertungsziele der Interaktionsszenarios auf (siehe ASL-Tabelle).

Architektonische Unterstützung von Usability			
Interaktionsszenario	Einfluss auf Usability, Komplexität von architektonischen Änderungen (ASL 0)	ASL 1 (erste Wiederholung von SATURN)	ASL 2 (zweite Wiederholung von SATURN)
Observing System State	(2, 2)	(0, 0)	(0, 0)
Abbrechen	(2, 2)	(1, 0)	(0, 0)
Using Applications Concurrently	nicht evaluiert	(0, 0)	(0, 0)
...	...	...	...

Tabelle 22: Architektur-Support-Level-Tabelle (ASL-Tabelle)

Im Beispiel der Tabelle 22 wurden im ersten Analysedurchgang die ersten zwei Interaktionsszenarios Oberserving System State und Abbrechen evaluiert. Beide wer-

den nicht architektonisch unterstützt und erfordern voraussichtlich komplexe Architekturänderungen. Zielvorgaben für die Analyse der nächsten beiden Versionen der Architektur (ASL 1 und ASL 2) zeigen, dass erst die zweite nachfolgende Version alle Interaktionsszenarios unterstützen soll. Beim zweiten Interaktionsszenario (Abbrechen) müssen nicht alle darin genannten Bedingungen erfüllt werden (z. B. wenn Prozesse abgebrochen werden können, aber noch nicht so schnell, wie spezifiziert).

ARCHITEKTURENTSCHEIDUNGEN AUFLISTEN Die Ergebnisse sollen die folgende konstruktive Arbeit vorbereiten. Dafür werden zuerst alle ermittelten szenariobeeinflussenden Architekturentscheidungen fortlaufend aufgelistet. Dies ermöglicht einen ersten Überblick über alle Analyseergebnisse, sortiert nach Interaktionsszenarios.

THEMEN BESCHREIBEN Die Ergebnisse sind nun sehr detailliert, wobei es möglich und wahrscheinlich ist, dass sich mehrere Architekturentscheidungen auf ein architektonisches Element, eine Verbindung zwischen architektonischen Elementen beziehen oder andere Gemeinsamkeiten aufweisen.

Um eine bessere fachliche Übersicht zu gewinnen, werden nun inhaltlich zusammenhängende Architekturentscheidungen als ein Thema agglomeriert. Ziel ist es, den Abstraktionsgrad zu erhöhen und den architektonischen Einfluss auf die Usability nachvollziehbarer offenzulegen; zudem sollen alle Erkenntnisse aufgeführt werden können, ohne alle ermittelten Details nennen zu müssen. Diese sind dann in den jeweiligen Evaluationsformularen nachlesbar. Das Schema in Tabelle 23 zeigt, wie die Themen strukturiert erörtert werden.

Element	*Beschreibung*
Titel	aussagekräftig, beispielsweise gemeinsame architektonische Elemente (z. B. Kommunikationskomponente), Interaktionen (z. B. Antwortverhalten) oder Anforderungen (z. B. Erweiterbarkeit)
Architekturentscheidungen	Architekturentscheidungen zu dem Thema auflisten
Erläuterung	Struktur und Verhalten der architektonischen Elemente mittels der Architekturentscheidungen kurz erklären
Voraussichtliche Unterstützung der Usability	kurz erklären und begründen, wie die Usability architektonisch gefördert wird
Hinweise auf mögliche Probleme bei der Benutzung	Entstehung potenzieller Usability-Probleme kurz erklären
Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen	(+) für vorteilhafte, (-) für nachteilige Wechselwirkungen mit anderen Qualitätsmerkmalen; diese können anschließend mit entsprechenden Inspektions- oder Testmethoden evaluiert werden
Mögliche Modifikationen	Auflistung und kurze Erläuterungen von Vorteilen von alternativen Architekturentscheidungen; konstruktive Aufgaben folgen der Analyse, daher hier nur konstruktive Hinweise für Architekten; Änderungskomplexität noch einmal schätzen und nennen

Tabelle 23: Schema für Themen

SONSTIGES In den Evaluationsformularen genannte offene Fragen und Kommentare sind hier aufzuführen.

4.2.7 *SATURN-Phase 5: Retrospektive*

Um die Methode und ihre Werkzeuge kontinuierlich zu verbessern, sollen die Anwender der Methode in der Retrospektive einschätzen, inwiefern sie Methode und Interaktionsszenarios so belassen oder verbessern würden.

BEFRAGUNG Mündlich oder schriftlich beantworten die Anwender der Methode die folgenden Fragen:

- Machbarkeit und Prozessunterstützung: Inwiefern war die Methode nützlich?
- Verständlichkeit: Inwiefern war die Methode verständlich? Gab es Missverständnisse über das Vorgehen oder die Werkzeuge der Methode?
- Reifung der Methode: Was soll an der Methode verbessert werden, was soll so bleiben?

DISKUSSION DER INTERAKTIONSSZENARIOS Abschließend wird diskutiert, welche Interaktionsszenarios neu erstellt oder überarbeitet werden sollen.

METHODEN DURCHFÜHREN Parallel oder nacheinander werden Softwarearchitekturanalysemethode und Nutzertest durchgeführt.

Das Evaluationsszenario eines Nutzertests besteht aus einer oder mehreren Teilaufgaben, die von Testpersonen durchgeführt werden müssen; diese Teilaufgaben integrieren einzelne Interaktionen, so wie sie von den Interaktionsszenarios von SATURN beschrieben werden.

VERGLEICH DER ERGEBNISSE Von der User-Evaluation ermittelte Probleme und Architekturentscheidungen werden tabellarisch gegenübergestellt und somit aufeinander abgebildet. Der Vergleich der Ergebnisse von Softwarearchitekturanalyse und Usability-Evaluation führt zu einer der vier Situationen, die in Tabelle 24 aufgeführt sind.

Typ	*Problem gefunden durch*	*Beschreibung*
1	SAA	SAA fand ein Problem, das nicht in der UE ermittelt wurde. Beispiel: da in der UE nur eine kleine Anzahl von Endgeräten getestet wurde, wurde nicht ermittelt, dass die Anwendung nicht darauf ausgelegt ist, tatsächlich alle festgelegten Endgeräte zu unterstützen. In der SAA wurde ermittelt, dass Endgeräteprofile nicht erweitert werden können. *Fragen an die UE: Wie wirkt sich das Problem auf die Nutzung aus?*
2	UE	Die User-Evaluation fand ein Problem, das nicht in der SAA ermittelt wurde. Beispiel: Die SAA ermittelte eine sehr gute Antwortzeit für Nutzerinteraktionen. Die Nutzer empfanden das Scrolling aber als zu schnell, da sie den Text nicht scannen konnten. *Fragen an die SAA: Wie kann das Problem behoben werden? Das Interaktionsszenario ist prinzipiell erneut in die nächste SAA aufzunehmen. Es ist zu prüfen, wie und auf welcher Architekturebene das Problem technisch behoben werden kann.*
3	SAA und UE	SAA und UE fanden ein Problem. In der SAA wurde durch Befragung des Architekten ermittelt, dass der Statusindikator des User Interfaces nacheinander Interaktionen zeigt, die bearbeitet werden. Diese entsprechen aber nicht dem Status des angeschlossenen Interaktionsgerätes. Die UE ermittelte, dass Nutzer durch die Statusangaben verwirrt waren: sie hatten vermutet, dass der Status des Interaktionsgerätes angezeigt wird. *Fragen an die SAA: Sind Problem und Ursachen damit erkannt und erklärt? Wie wird das Problem behoben und bis wann?*
4	keine	Weder SAA noch UE ermittelten ein Problem.

Tabelle 24: Vergleich der Ergebnisse von SA-Analyse (SAA) und User-Evaluation (UE)

Die ersten drei Situationen (Typ 1, 2, 3) treten auf, während die Methoden durchgeführt werden. Die vierte Situation (Typ 4) kann nicht evaluiert werden; der Fall zeigt aber, dass auch ein Methodenmix nicht alle möglichen Probleme aufdecken kann.

Die Situation von Typ 1 betrifft Interaktionen, die während der Softwarearchitekturanalysemethode (SAA) überprüft werden können, aber nicht über eine User-Evaluation: beispielsweise wegen des Studiendesigns oder aus technischen Gründen.

Die Situation von Typ 2 betrifft Usability-Probleme, die mit der User-Evaluation, aber nicht durch die Softwarearchitekturanalysemethode aufgedeckt wurden. Die Ursache solcher Usability-Probleme liegt meist in menschlichen Fehlern und Erkennungsproblemen.

In der Situation von Typ 3 wird bei der SAA eine Architekturentscheidung ermittelt, die ein Usability-Problem verursacht, das in einer User-Evaluation ermittelt wurde.

Die Ergebisse werden aufeinander abgebildet und diskutiert. Mögliche Lösungen und Erklärungen für die Probleme sind zu identifizieren, ungelöste Probleme (sofern notwendig) zu markieren. Schließlich werden die offenen Punkte auf der Liste diskutiert. Offene Punkte sind Probleme, deren Ursache oder Effekte nicht klar und eindeutig identifiziert werden konnten. Am Ende der Analyse können die einzelnen Probleme und szenario-beeinflussenden Entscheidungen mit einer der folgenden Statusinformationen annotiert werden:

- Akzeptiert: Die Evaluatoren sind sich über eine bestimmte Problemquelle und dessen Lösung einig.
- Weitere Evaluation nötig: Offene Punkte wurden identifiziert, aber weitere Evaluationen sind notwendig, um ein Problem zu klären und eine angemessene und passende Lösung zu finden.
- Neues Problem identifiziert: Entweder die Usability-Evaluation oder die SAA identifizierte Probleme. Daraus ergeben sich weitere Evaluationen.
- Problem identifiziert, keine Lösung gefunden: Eine Lösung konnte nicht erreicht werden. Abhängig von den Resultaten sind dann entweder eine erneute User-Evaluation oder Softwarearchitekturanalyse sinnvoll. In beiden Fällen werden weitere Diskussionen über die offenen Fragen initiiert.

Verglichen mit den Ergebnissen einer einzelnen Methode sollen durch den Methodenmix mehr und fundiertere Erkenntnisse darüber ermittelt werden, wie gut die Usability einer Anwendung ist und inwiefern sie verbessert werden kann.

1. *Wie kann der Nutzungskontext in eine szenario-basierte Architekturanalysemethode integriert werden?*

   Der Nutzungskontext bestimmt die Usability-Anforderungen einer konkreten Anwendung (siehe Usability-Definition, S. 1). Er beschreibt in erster Linie zu beachtende Rahmenbedingungen über Umgebung, Nutzer, Endgeräte und die mobile Anwendung und insbesondere die Aufgaben, die mit der Anwendung möglich sein sollen. Diese Aufgaben werden durch Interaktionen mit der Anwendung unterstützt und eignen sich zur Analyse, da sie leicht in Szenarios umgewandelt werden können.

   Damit der Nutzungskontext in der Methode von Architekten beschrieben werden kann, wird eine Literaturstudie durchgeführt, in der typische Parameter von Nutzungskontextfaktoren von mobilen Anwendungen ermittelt werden. Diese werden tabellarisch zusammengestellt. Die Teilnehmer können sie als Checkliste nutzen und die Situation verstehen, in der die Nutzer mit der Anwendung interagieren. Insbesondere die Aufgaben, aber auch die anderen Rahmenbedingungen führen zu konkreten Interaktionen zwischen Nutzer und System. Diese Interaktionen sind in Interaktionsszenarios überführt.

   In SATURN wird Usability also operationalisiert, indem benutzer- oder systeminitiierte Interaktionen und somit die erwartete Funktionalität beschrieben werden. Damit werden Rahmenbedingungen für die Benutzung der Anwendung in überprüfbare funktionale Anforderungen und Qualitätsanforderungen transformiert.

2. *Welches Format eignet sich dafür, Usability-Anforderungen so zu beschreiben, dass sie für die Architekturanalyse ausreichend viele Informationen enthalten und als Kriterien überprüfbar sind?*

   Es ist notwendig, ein Soll-Verhalten zu definieren, welches als Kriterium überprüft werden kann. Das Format der Szenarios von ATAM bildet diese Anforderung ab, nutzt aber Usability-Attribute. Der Umweg über die Usability-Attribute soll aber vermieden werden. Dies ist möglich, wenn als Stimulus nicht ein Qualitätsattribut, sondern eine konkrete Interaktion beschrieben wird. Die Interaktionsszenarios der Methode SATURN beschreiben deshalb benutzer- oder systeminitiierte Interaktionen und die von den Nutzern erwartete Reaktion des Systems.

3. *Wie können Personen, die keine Usability-Experten sind, dabei unterstützt werden, das Qualitätsmerkmal Usability in Szenarios zu operationalisieren?*

   In SATURN-Phase 1 werden Interaktionen der Anwendung ermittelt, in SATURN-Phase 2 werden bestimmte Interaktionen ausgewählt und in Interaktionsszenarios umgewandelt. Dieses schrittweise Vorgehen wahrt das Abstraktionsniveau jeder Phase und vereinfacht dadurch die jeweilige Aufgabe, da nicht sofort Details fokussiert werden. Außerdem können die Anwender der Methode unvoreingenommen ihre eigenen Ziele und Wünsche festhalten, bevor sie im Katalog der Interaktionsszenarios recherchieren und dort ihren Fokus auf das Vorhandene einschränken. Die im Anhang enthaltenen 50 Interaktionsszenarios dienen als Vorlagen und als Beispiele (siehe Anhang A.2, S. 204 ff.) für eigene Interaktionsszenarios.

4. *Welche Kriterien eignen sich zur Auswahl der Szenarios? Wie kann methodisch integriert werden, dass die Auswahl fachlich motiviert erfolgt?*

   Die Selektion von Interaktionsszenarios ist in SATURN fachlich getrieben, denn für den wirtschaftlichen Erfolg ist es notwendig, dass die Hauptinteraktion (bei

einer Suchmaschine das Suchen) und das Alleinstellungsmerkmal gegenüber der Konkurrenz (bei einer Suchmaschine die Algorithmen, die Suchergebnisse, die Darstellung und mögliche Interaktionen mit den Suchergebnissen) architektonisch unterstützt werden. Deshalb sind Interaktionsszenarios für den Hauptzweck der Anwendung und das Alleinstellungsmerkmal vorausgewählt.

In SATURN-Phase 2 bestimmen die Anwender dieser Methode in einer fachlichen, offenen und dokumentierten Diskussion, ob ein Interaktionsszenario ausgewählt werden sollte. Um Transparenz zu schaffen, werden Argumente, die für ein Interaktionsszenario gesprochen haben, in die Evaluationsformulare aufgenommen, ebenso, wer es favorisiert. Der Einfluss anderer Gründe als fachlicher kann nicht ausgeschlossen werden, wird aber durch die methodische Vorgabe verringert.

5. *Wie können die Hilfsmittel der Analyse so erstellt werden, dass die Anwender der Methode nicht alle schon existierenden Lösungen (Patterns) kennen oder über ein sehr umfangreiches Fachwissen verfügen müssen?*

   Die Methode SATURN erfordert es nicht, Patterns zu kennen, denn die Analyse folgt einer anderen Fragestellung: es geht nicht darum, eine bestehende Architektur mit Patterns zu vergleichen, sondern es geht darum, eine bestehende Architektur daraufhin zu untersuchen, wie Architekturentscheidungen ein konkretes Interaktionsszenario unterstützen.

   Für den optionalen, konstruktiven Teil der Methode können Patterns notwendig werden: so eignen sich die Interaktionsszenarios dazu, Lösungsräume zumindest einzugrenzen, da jedes Interaktionsszenario mindestens die Patterns referenziert, von denen es abgeleitet ist. Es ist zudem möglich und sinnvoll, weitere Patterns sowie eigene Projekte als Referenzen zu ergänzen. Der Lösungsraum eines Interaktionsszenarios ist somit eingeschränkt; Anwender von SATURN können sich wenige Patterns anschauen.

6. *Welche Fragetechnik verringert die Freiheitsgrade der Analyse und legt ihr Vorgehen offen?*

   In SATURN werden standardisierte Fragen gestellt, die dazu führen, dass das zu betrachtende Problem (die architektonische Unterstützung eines Interaktionsszenarios) schrittweise eingrenzt und die entsprechenden Lösungen erfragt werden. Die Interaktionsszenarios werden stichwortartig mit Nutzfällen genauer spezifiziert. Welche Komponenten und Schnittstellen der Softwarearchitektur unterstützen, wird ermittelt. Dieser analytische Teil führt zu einem Verständnis über die aktuelle Architektur.

7. *Wie kann die architektonische Unterstützung von Usability bestimmt und bewertet werden?*

   Ein Interaktionsszenario beschreibt eine konkrete Interaktion sowie ihre erwartete Qualität und damit eine Usability-Anforderung an die betreffende mobile Anwendung. Wenn ein Interaktionsszenario nicht oder nur teilweise unterstützt wird, entsteht voraussichtlich ein Usability-Problem. Denn dann werden bestimmte Interaktionen, die auf eine bestimmte Art und Weise möglich sein sollen, nicht oder nur teilweise durchführbar sein.

   Es wird eingeschränkt, dass die architektonische Unterstützung von bestimmten Usability-Anforderungen analysiert wird. Die Usability selbst kann nur bei der Benutzung der Anwendung ermittelt werden.

   Stufen werden festgelegt, die beschreiben, inwiefern die Usability-Anforderung, die ein Interaktionsszenario beschreibt, umgesetzt wurde (Tabelle 26).

Typ	*Beschreibung*	*Einfluss auf Usability*
0	Unterstützt das Interaktionsszenario	Response und Response-Prüfung werden unterstützt; kein negativer architektonischer Einfluss auf die Usability ersichtlich
1	Unterstützt das Interaktionsszenario teilweise	Response und/oder Response-Prüfung werden teilweise unterstützt; kleinerer negativer architektonischer Einfluss auf Usability identifiziert
2	Unterstützt das Interaktionsszenario nicht	Response und Response-Prüfung werden nicht unterstützt; großer negativer Einfluss auf Usability identifiziert
3	Unterstützt das Interaktionsszenario teilweise oder nicht	unbekannt, inwiefern Response und Response-Prüfung nicht unterstützt werden; architektonischer Einfluss auf Usability kann von Typ 1 oder 2 sein, nicht jedoch von Typ 0

Tabelle 25: Einfluss einer Architekturentscheidung auf Usability

Ein Interaktionsszenario von Typ 1, 2 oder 3 wird nicht architektonisch unterstützt. Es ist also notwendig, sofort oder später alternative Lösungen zu erarbeiten.

Dies ist ein konstruktiver Anteil der SAA, der notwendig ist, um zu bestimmen, inwiefern ein Interaktionsszenario architektursensitiv ist. Tabelle 26 beschreibt, wie die Architektursensitivität bestimmt wird.

Typ	*Beschreibung*	*Änderungsaufwand*
0	Keine Modifikationen	keine Änderungen notwendig
1	Einfache Modifikationen	Verhalten einer oder mehrerer Komponenten und/oder Schnittstellen soll geändert werden; Modifikationen erfordern keine strukturellen Änderungen
2	Komplexe Modifikationen	Ein oder mehrere architektonische Elemente müssen hinzugefügt, entfernt oder anders kombiniert werden; Modifikationen betreffen sowohl Struktur als auch Verhalten der architektonischen Elemente
3	Unvorhersehbare Komplexität	Unbekannter Aufwand, kann von Typ 1 oder 2 sein, nicht jedoch von Typ 0

Tabelle 26: Komplexität von architektonischen Änderungen

Ist es nicht möglich, Vorschläge zu unterbreiten, ist das als Typ 3 zu vermerken. Diese Art der Bewertung zielt darauf ab, dass Architekten die Architektur analysieren und gleichzeitig erkennen, wo Handlungsbedarf besteht.

8. *Wie kann der iterative und zyklische Prozess zum Entwurf der Architektur methodisch unterstützt werden?*

   Jedes Interaktionsszenario soll unterstützt werden und nicht architektursensitiv sein. Ob die Zielbewertung „die Anforderung wird umgesetzt, keine Modifikationen nötig“ erreicht wird, fasst die „Architektur-Support-Level-Tabelle“ zusammen (siehe Abbildung 27).

Architektonische Unterstützung von Usability						
Interaktionsszenario	Produkt- oder Nutzungskontext	CASSINI	Beginn der Evaluation	Einfluss auf Usability, Softwarearchitektur (ASL 0)	ASL 1 (erste Wiederholung)	ASL 2 (zweite Wiederholung)
Observing System State	✓	–	0	2, 2	0, 0	0, 0
Abbrechen	✓	–	0	2, 2	0, 0	0, 0
Nachsichtiges Format	–	✓	0	2, 1	2, 1	0, 0
...	...	...	...	...	...	...

Tabelle 27: Architektur-Support-Level-Tabelle mit Bewertungen und Prioritäten der Interaktionsszenarios

Die Zielvorgaben der Interaktionsszenarios werden zusammengefasst und nach dem Interpretieren der Resultate noch einmal überarbeitet und festgelegt. Die Interpretation ist eine inhaltlich standardisierte Diskussion übergreifender Themen und sich daraus ergebender Anforderungen bzw. offene Fragen an Stakeholder, Architekten und Usability Engineering Experten.

Schema für die Beschreibung von Themen:

- Der Titel eines Themas kann beispielsweise gemeinsame Ursachen betreffen (z.B. Probleme mit einem Kommunikationsmodul), ähnliche Interaktionen (z.B. Antwortverhalten, Behandlung von Formulareinträgen) oder Anforderungen (z.B. Erweiterbarkeit).
- Alle zu einem Thema gehörenden Entscheidungen werden in einer Tabelle aufgelistet.
- Das Thema wird, die Inhalte der Tabelle referenzierend, in einem Absatz beschrieben.
- Beschrieben werden Usability-Probleme und die Ableitung der Fragestellung an Usability-Experten, z.B. neue Anforderungen, Fragen an Benutzertests.
- Beschrieben werden nötige Softwarearchitektur-Modifikationen, mögliche Alternativen, Ideen zur Behebung oder offene Fragen. Fragestellungen an Architekten (z.B. hinsichtlich der Anforderungen, Fragen an Stakeholder) werden abgeleitet.
- Beschrieben werden Wechselwirkungen mit anderen Qualitätseigenschaften in einer Liste mit Aufzählungszeichen (+) für vorteilhafte Beziehungen, (-) für nachteilige Beziehungen. Bestehende oder vermutete Wechselwirkungen können im Anschluss mit den entsprechenden Inspektions- oder Testmethoden evaluiert werden.

Usability wird in Usability-Anforderungen operationalisiert, die als Interaktionsszenarios dargestellt werden. Diese Interaktionsszenarios stehen fast immer im Fokus. Wenn abschließend übergreifende Themen diskutiert werden, spielen die betreffenden Interaktionsszenarios kaum eine Rolle, sondern die ermittelten

Architekturentscheidungen. Der Bogen wird allerdings am Schluss wieder geschlagen, indem die Zielvorgaben der Interaktionsszenarios in der Architektur-Support-Level-Tabelle (Abbildung 27) noch einmal überprüft und aktualisiert werden.

Das Ergebnis von SATURN sind neben Analyse und Zielvorgaben gewünschte Modifikationen und offene Fragen.

Diese Ergebnisse stellen neue Eingaben für die beteiligten Prozesse dar. Die Architekten prüfen beispielsweise neue Ideen, neue Anforderungen und offene Fragen für Stakeholder. Die Usability-Experten erhalten beispielsweise neue Anforderungen, Fragen für Benutzertests, Rechercheaufträge und Rahmenbedingungen für das User Interface Design.

9. *Wie kann die interdisziplinäre Kooperation zwischen den beteiligten Prozessen unterstützt werden?*

   Die Methode SATURN wurde so erstellt, dass sie mit dem Usability Engineering koordiniert werden kann. Eine sinnvolle Voraussetzung für eine Kooperation von Prozessen ist, dass die Ausgangslage (Eingaben) gleich und Ausgaben vergleichbar sind. Der Nutzungskontext und Usability-Anforderungen bilden die gemeinsame Ausgangslage, die Ausgaben sind die ermittelten (potenziellen) Usability-Probleme.

   Wenn der Nutzungskontext beschrieben ist, kann er in die Methode übernommen werden, sonst wird er erstellt und kann vom Usability Engineering weiterverwendet werden. Das gilt analog für Usability-Anforderungen und Interaktionsszenarios. Die Ergebnisse der Methoden sind jeweils für beide Gruppen verwendbar. Eine User-Evaluation ermittelt Usability-Probleme. Ergebnisse einer User-Evaluation können, als Interaktionsszenarios beschrieben, in die Architekturanalyse einfließen. Wie SATURN mit einer User-Evaluation verbunden werden kann, wurde als Methode beschrieben (siehe Abschnitt 4.3, S. 74ff.) und erfolgreich in einer Fallstudie erprobt (Abschnitt 5.2.8, S. 111 ff.).

   SATURN ermöglicht einen Informationsaustausch zwischen der Erstellung einer Softwarearchitektur und dem Usability Engineering, etabliert geeignete Kommunikationsprozesse und ergänzt eine Methode zur Abstimmung arbeitsteiliger Aktivitäten.

Nachdem die Methode beschrieben und die Forschungsfragen betrachtet wurden, stellen die folgenden Tabellen 28 und 29 den Vergleich der früheren Methoden mit der neuen Methode SATURN dar.

Elemente	*SALUTA*	*ATAM*	*SATURN*
Definition von Kontext und Qualitätsmerkmalen	(+) Usability im Mittelpunkt (+) Nutzungskontext beachtet	(+) Usability als ein Qualitätsmerkmal unter vielen (+) Umfassende Bestimmung des Kontexts (o) ohne Nutzungskontext, kann aber ergänzt werden	(+) Usability im Mittelpunkt (+) Umfassende Bestimmung des Nutzungskontexts (+) Austauschbeziehungen mit anderen Qualitätsmerkmalen werden betrachtet
Bestimmung und Selektion von Szenarios	(-) Tripel mit geringem Informationsgehalt (-) Mehrere Szenarios für eine Interaktion (-) Bildung einer Rangfolge von Usability-Attributen, die aber nominal skaliert sind und für jede Interaktion gelten (-) freie Auswahl ohne Anleitung	(+) ausführliche Beschreibung (-) Usability-Attribute als Ausgangspunkt für konkrete Anforderungen (-) Auswahl der Szenarios ist leicht beeinflussbar und wird nicht fachlich begründet	(+) konkret definiert (+) Format zwingend (+) Auswahl fachlich; nachvollziehbar belegt
Unterstützung der Analyse	(-) Dokumentation selbstbestimmt (+) Wissensbasis aus Usability-Patterns und Usability-Properties	(+) ausführlich durch Anleitung (+) viele Vorlagen (+) Wissensbasis aus Usability Supporting Architectural Patterns und Taktiken	(+) ausführlich durch Anleitung (+) viele Vorlagen (+) Wissensbasis aus Interaktionsszenarios, die von Patterns abgeleitet sind und auf diese verweisen

(+) zweckdienlich, [o] ausreichend, (-) ungenügend hinsichtlich der Ziele dieser Arbeit

Tabelle 28: Bewertung der Methoden ATAM, SALUTA und SATURN (Teil 1/2)

Elemente	*SALUTA*	*ATAM*	*SATURN*
Bestimmen der Analyseergebnisse	(-) Prüfung, ob jede Usability-Property und jedes Usability-Pattern jedes Szenario unterstützt (-) Bewertung in freier Diskussion mit maximalem Freiheitsgrad (-) nicht gut nachvollziehbar, da Dokumentation selbstbestimmt (-) Rückverfolgbarkeit nicht gewährleistet, da zu viele Freiheiten	(-) Analysten stellen Fragen, die aber unbekannt sind, weil sie von deren Wissen und Erfahrung abhängen (-) wie genau die Analyse geschieht, ist unbekannt (+) gut dokumentiert (-) nicht rückvollziehbar, da Bezug zu einzelnen Szenarios während der Analyse verloren geht	(+) Vereinfachtes und standardisiertes Vorgehen: Interaktionsszenarios werden in Nutzfälle verfeinert, zu denen die architektonischen Elemente ermittelt werden (+) nachvollziehbar (+) rückvollziehbar
Zusammenspiel von Prozessen	(+) Informationsfluss vom Usability Engineering (-) kein Informationsfluss zum Usability Engineering	(+) Informationsfluss vom Usability Engineering (-) kein Informationsfluss zum Usability Engineering	(+) Informationsfluss vom Usability Engineering (+) Informationsfluss zum Usability Engineering (+) Kooperation durch gemeinsame Evaluation möglich (+) Koordination, wenn z. B. Nutzungskontext gemeinsam erarbeitet und verwendet wird

(+) zweckdienlich, [o] ausreichend, (-) ungenügend hinsichtlich der Ziele dieser Arbeit

Tabelle 29: Bewertung der Methoden ATAM, SALUTA und SATURN (Teil 2/2)

4.5 ZUSAMMENFASSUNG

In diesem Kapitel wurde die Methode SATURN erstellt und präsentiert. In der ersten Phase werden einfache Usability-Techniken durchgeführt, um den Nutzungskontext zu erarbeiten und damit die Auswahl der Interaktionsszenarios vorzubereiten. In der zweiten Phase werden Interaktionsszenarios aus dem Katalog von 50 Interaktionsszenarios ausgewählt; mit einer Vorlage können neue Interaktionsszenarios auf einem vergleichbaren Abstraktionsniveau geschaffen werden; die Auswahl der Interaktionsszenarios erfolgt nach fachlichen Kriterien. In der dritten Phase werden die Interaktionsszenarios evaluiert, um Architekturentscheidungen zu ermitteln und zu bewerten. In der vierten Phase werden die Architekturentscheidungen thematisch neu gruppiert und als Themen zusammenfassend beschrieben. In der fünften Phase findet ein Rückblick auf die Durchführung hinsichtlich der Machbarkeit, der Prozessunterstützung, der Verständlichkeit sowie der Verbesserung der Interaktionsszenarios

statt. Zusätzlich zur Methode SATURN wurde ein Vorgehen beschrieben, wie diese als Experteninspektion mit einem Nutzertest kombiniert werden kann.

Das folgende Kapitel 5 behandelt die Fallstudien, in denen die Methode SATURN zweimal und die Kombination mit einem Nutzertest einmal erprobt wurde.

DANKSAGUNGEN

Die erste Version der Methode erschien in den Proceedings der ECSA/WICSE 2009 [BG09]; Fallstudien wurden veröffentlicht in [BGG10, BG10a]. Eine vorläufige Version der Abschnitte zu Mobilität und Nutzungskontext erschienen im *Journal of Systems and Software* mit Thomas Grill und Volker Gruhn [BGG10]. Vielen Dank an alle Gutachter und Kommentatoren. Vielen Dank an Andreas Fiesser, der eine Pattern-Version dieser Methode prüfte. Herzlichen Dank an meine Mitautoren Thomas Grill, Stefan Seitz und Volker Gruhn: Volker Gruhn für seine punktgenauen Reviews, Stefan Seitz für seine Arbeiten zu den Interaktionsszenarios sowie deren Auswahl [Sei09] und besonders Thomas Grill für die gemeinsame Forschungsarbeit zur Koordination mit dem Usability Engineering.

5 FALLSTUDIEN

Abschnitt 5.1 erläutert das Studiendesign. Danach behandelt Abschnitt 5.2 eine Fallstudie im Rahmen eines Forschungsprojektes, in dem SATURN durchgeführt und mit einem Nutzertest kombiniert wurde; gefolgt von einer Fallstudie aus der Praxis (Abschnitt 5.3). Abschnitt 5.4 fasst das Kapitel zusammen.

5.1 STUDIENDESIGN

5.1.1 *Canonical Action Research*

Literaturstudien, Fallstudien und Action Research sind typische Forschungsmethoden der Method Construction [BWHW05]. Das heißt, eine Methode kann zuerst mittels Literaturstudien konstruiert und danach praktisch erprobt und schließlich immer weiter verfeinert werden.[1]

Nach [CKK02, BG04, BLBV04] werden szenario-basierte Softwarearchitekturanalysemethoden mit Fallstudien validiert. In Fallstudien wird ein einzelnes Ereignis (oder wenige Ereignisse) systematisch untersucht. Ziel einer solchen, in die Tiefe gehenden Analyse und damit qualitativen Forschung [Ber09] ist es, ein Ereignis genau zu beschreiben, zu erklären und zu verstehen [Bro90, BB03]. Allerdings besteht bei Fallstudien das Risiko, die ursprünglichen Ziele durch Richtungsänderungen zu invalidieren. Um diesem entgegen zu wirken, ist es sinnvoll, die Fallstudie in einen rigorosen Forschungsprozess einzugliedern, der eine planvolle Vorbereitung sowie eine kritische Reflexion ermöglicht und dabei die Kooperation von Teilnehmern und Forschern fördert [ALMN99]. Kanonische anwendungsnahe Forschung (Canonical Action Research, CAR) bildet einen Rahmen, in den Fallstudien eingebettet werden können.

Action Research (AR, anwendungsnahe Forschung) ist keine Forschung im Labor, sondern Forschung im realen Umfeld. In AR verbinden sich Forschung und Praxis miteinander: so befasst sich anwendungsnahe Forschung mit Forschungsfragen, in denen sowohl Probleme als auch deren Lösungsprozess relevant sind [DMK04]. Forscher beziehen deshalb die Mitglieder einer Organisation als Teilnehmer und Teilhabende hinsichtlich der Forschungsziele ein [ALMN99, ESSD07]. Um Action Research rigoros zu gestalten, entwarfen Davison, Martinsons und Kock [DMK04] die Forschungsmethodik des Canonical Action Research.

Die fünf Prinzipien des CAR [DMK04] erfordern:

1. Vereinbarung zwischen Forscher und Klienten: im Vorfeld der Fallstudie von den beteiligten Parteien darüber, wie das Forschungsprojekt abläuft, welche Ziele beide Parteien verfolgen, wie Rollen und Verantwortlichkeiten untereinander aufgeteilt sind und wie Daten gesammelt werden;

2. zyklisches Vorgehen: zyklisches Prozessmodell (siehe Abbildung 17) soll eingehalten werden;

3. theoretische Grundlagen: notwendig für Aktivitäten im Projekt, betrachtete Probleme, deren Erklärungen, Interventionen sowie Reflexion;

4. Änderungen durch Aktion: Änderungen der Methode gemeinsam von Forscher und Klient erwünscht, fachlich motiviert, bewertet und dokumentiert;

1 Tabelle 37 auf Seite 168 geht genauer auf Ziele, Methoden, Validationsziele eines dreistufigen Forschungsprozesses ein.

5. Lernen durch Reflexion: Lerneffekte schaffen und festzuhalten, durch gemeinsamen Rückblick auf das Forschungsprojekt sowie die Dissemination der Erkenntnisse.

5.1.2 *Canonical Action Research und SATURN*

Das Forschungsprojekt zur Methode SATURN und die folgenden Abschnitte untergliedern sich nach dem zweiten Prinzip, also für jede Fallstudie in Diagnose, Planung, Intervention, Beurteilung und Reflexion. Abbildung 17 zeigt einen CAR-Zyklus.

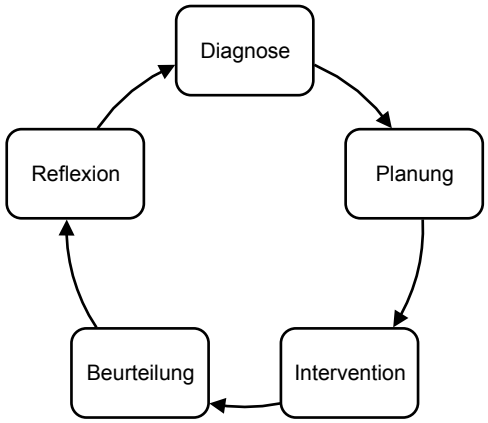

Abbildung 17: Zyklus der Anwendungsnahen Forschung nach [DMK04]

DIAGNOSE (EINGANG) Zuerst wird die Ausgangslage definiert. Thematisiert werden der Reifegrad der Methode und die mobile Anwendung selbst.

PLANUNG Nach dem ersten CAR-Prinzip wird vereinbart, wie Zeitplanung, Ziele, Rollenverteilung und Datenerhebung erfolgen. Die Forscher, Analysten und Architekten planen, welche Personen beteiligt sind, welche Aufgaben diese übernehmen, welche personellen und zeitlichen Aufwände zu erwarten sind, welche Ziele mit der Analyse verfolgt werden, wo sie stattfindet, wie welche Daten gesammelt und ausgewertet werden und ob die Veröffentlichung der Ergebnisse eingeschränkt ist.

INTERVENTION („DURCHFÜHRUNG“) Zu Beginn der Methode werden bei einem Kurzvortrag die theoretischen Grundlagen für die Methode und ihre Werkzeuge vorgestellt, d. h. verwendete Begriffe und Methoden des Usability Engineerings, des Entwurfs der Softwarearchitektur, der Zusammenhang zwischen Usability und Softwarearchitektur und Wechselwirkungen zwischen Qualitätsmerkmalen (siehe Kapitel 4, 255ff.). In den entsprechenden Kapiteln finden sich die theoretischen Grundlagen zur Methode SATURN (Kapitel 4, S. 48) und zu den Interaktionsszenarios (Kapitel 3, S. 27). Das dritte CAR-Prinzip wird somit berücksichtigt.

Während der Fallstudien werden Zwischenergebnisse der Methode, Kommentare der Teilnehmer und die Dauer der einzelnen Arbeitsabschnitte erhoben. Die Datensammlung innerhalb der Methode erfolgt dabei mit direkten und unabhängigen Techniken: SATURN setzt direkte Fragetechniken ein, ebenso ein Brainstorming sowie einen Walkthrough mit der „Think Aloud“ Methode und abschließend eine mündliche Befragung mit offenen Fragen. Ergänzt wurden diese Informationen um die unabhängige Datenerhebung durch eine rückblickende Analyse der Unterlagen und Kommentare.

Gemäß des vierten CAR-Prinzips werden Aktivitäten der Methode schon während der Durchführung einer Fallstudie geändert, sofern dies beide Parteien als sinnvoll erachten.

BEURTEILUNG Im Anschluss an die ersten vier SATURN-Phasen werden in einer fünften Phase Machbarkeit und Verbesserungspotenziele gemeinsam betrachtet; darauf basierend sollen Methode und Hilfsmittel rückblickend durch den Forscher überarbeitet werden (fünftes CAR-Prinzip). Inhaltlich betrifft das die Fragen:

- Machbarkeit: Konnte mit der Methode SATURN ermittelt werden, inwiefern die Interaktionsszenarios in der untersuchten Anwendung architektonisch unterstützt werden?
- Nützlichkeit: Hat es sich aus Sicht der Architekten/Analysten gelohnt, die Methode SATURN durchzuführen?
- Verständnis: Inwiefern traten Missverständnisse über das Vorgehen oder die Werkzeuge dieser Methode auf?

REFLEXION (AUSGANG) Nach den Fallstudien wird die Methode durch den Forscher rückblickend überarbeitet (ebenfalls fünftes CAR-Prinzip). Es wird festgestellt, inwiefern Methode und Hilfsmittel optimiert werden sollen (Aufwand-Nutzen-Relation) und welche Wechselwirkungen offengelegt wurden. Konkrete Veröffentlichungen werden mit den beteiligten Personen abgestimmt und vorbereitet.

5.1.3 *Kurzvorstellung der Fallstudien*

Um die Machbarkeit der Methode SATURN zu prüfen, wurden zwei Fallstudien durchgeführt, die dem typischen Fall entsprechen: analysiert wurden mobile Anwendungen, die auf einem mobilen Endgerät laufen und von mobilen Benutzern in mobilen Umgebungen verwendet werden.

Die erste Fallstudie prüfte die Gültigkeit der Methode unter Laborbedingungen: Die Anwendung „Shake Your Mac“ (SYM) repräsentiert die Software eines wissenschaftlichen Prototyps für neue Interaktionstechniken. Die Methode wurde damit erstmalig durchgeführt. Die zweite Fallstudie prüfte eine schon am Markt etablierte mobile Anwendung, „TrafficScanner“ (TS), mit der Autofahrer Staus auf Autobahnen einer Zentrale melden können. Die Methode wurde unter praktischen Gesichtspunkten erprobt.

5.2 FALLSTUDIE „SHAKE YOUR MAC“

5.2.1 *Diagnose*

Die erste Version der Methode SATURN und die ersten 20 Interaktionsszenarios lagen vor, um die Methode im Rahmen eines wissenschaftlichen Forschungsprojektes erstmalig zu erproben.

Die Anwendung „Shake Your Mac“ (SYM) ist ein wissenschaftlicher Prototyp, um neuartige Interaktionen in einer mobilen Umgebung zu testen. Wie es für mobile Anwendungen charakteristisch ist, läuft SYM auf mobilen Endgeräten, die drahtlos miteinander kommunizieren und von mobilen Nutzern eingesetzt werden. Ziel des Architekten von SYM war es, nach der Analyse eine neue Version des Prototypen zu entwickeln und zu ermitteln, inwiefern die Usability-Anforderungen schon von der ersten Version unterstützt werden.

5.2.2 *Planung*

Ein Analyst (Forscher) und der Architekt der Anwendung führten die Analyse durch, beide verfügen über Erfahrungen im Interaktionsdesign und in der Programmierung.

Der Architekt analysierte die Anwendung selbst, der Analyst achtete darauf, dass das Vorgehen eingehalten und dokumentiert wird.

Die Ergebnisse zu veröffentlichen, war ein Hauptziel der wissenschaftlichen Kooperation. Die Forschungsmethode des Action Research wurde thematisiert und ein aktives Hinterfragen sowie Offenlegen von Problemen vereinbart.

Nach der Untersuchung der architektonischen Unterstützung der Usability mit SATURN, wurde die Usability des Prototyps mit einem Nutzertest evaluiert.

Studiendesign, Durchführung und Auswertung der User Evaluation von Shake Your Mac erfolgten durch Thomas Grill; Studiendesign, Durchführung und Auswertung von SATURN erfolgten durch Bettina Biel. Die Verbindung beider Perspektiven war ein gemeinsames Forschungsprojekt zur Frage, wie Usability Engineering und Softwaretechnik besser miteinander kooperieren können; veröffentlicht wurde der Bericht dazu in [BGG10].

Die folgende Beschreibung folgt der ersten Version der Methode, deren wesentlicher Teil – die Argumentation von Interaktionsszenarios zu den Architekturentscheidungen und deren Bewertung – bestehen blieb. Die Unterlagen vermischen die englische und die deutsche Sprache; dies kann leider nicht im Nachhinein angepasst werden, da hier nur das tatsächliche Vorgehen abgebildet werden kann. Aus diesem Grund bleiben auch Ungenauigkeiten in den Analyseformularen erhalten.

5.2.3 *Durchführung der Fallstudie SYM: SATURN-Phase 1*

5.2.3.1 *Name und Verwendungszweck*

Die Anwendung Shake Your Mac (SYM) analysiert Daten, die es von Sensoren zur Beschleunigungsaufnahme (engl. accelerometer, acceleration sensor) des mobilen Endgeräts (Apple MacBook) erhält. Werden Daten als Interaktionen von Benutzern interpretiert, werden sie in Interaktions-Events übersetzt. Eine Komponente, die sich auf einem anderen Endgerät (Windows PC) befindet, kann sich mit dem mobilen Endgerät verbinden und die Interaktions-Events erhalten. An dieser Komponente sind weitere Anwendungen – Microsoft Powerpoint (PPT) und Google Earth (GE) – angemeldet, welche die Interaktions-Events erhalten und umsetzen.

5.2.3.2 *Nutzungskontext*

Zwei Benutzergruppen unterscheiden sich in Alter, Sichtvermögen und IT-Erfahrung und agieren in zwei Umgebungstypen, in denen die Anwendung verwendet wird: im Seminarraum bzw. in einem Museum. Um selbstgestellte Aufgaben mit einer Anwendung zu lösen, kann ein Nutzer mit dem Endgerät vorwärts und rückwärts navigieren sowie auf dem Bildschirm herein- und herauszoomen. Dafür ist es nötig, das Endgerät zu bewegen oder haptisch zu interagieren. Das Gerät kann beispielsweise geschüttelt oder gekippt werden und es ist möglich, auf oder neben das Gerät zu klopfen.

Bei einem Walkthrough durch den Prototypen wurde die Anwendung Google Earth gestartet, ein bestimmter Ort wurde gesucht sowie genauer betrachtet und danach wurde ein weiterer Ort gesucht. Danach wurde Google Earth beendet.

Folgende Usability-Anforderungen wurden danach für die Analyse der Softwarearchitektur notiert.

- Absicherung und die Automatisierung des Verbindungsaufbaus nach einem Verbindungsverlust
- Abbrechen von Interaktionen

Im anschließenden fünfminütigen Brainstorming wurden folgende weitere Usability-Anforderungen festgehalten, die durch die Anwendung unterstützt werden sollen.

- gutes Feedback für die Benutzer
- Ergänzen neuer Endgeräte, damit Nutzer mit einem Endgerät ihrer Wahl mit der Anwendung interagieren können (Anwendungsfall für das Museum)

Abbildung 18 zeigt die erstellte kurze Übersicht des Nutzungskontexts.

Umgebungen	Benutzer	Interaktionen und Aufgaben	Endgeräte	Anwendung
Museum (vielfältige Störungen durch soziale Interruptionen), durch Umgebungsgeräusche	Museumsbesucher, Familienmitglieder	Navigieren: vorwärts, rückwärts, zoomen usw.	für den Prototyp: aktuell Mac als Nutzerschnittstelle, muss bewegt, geschüttelt werden, um zu navigieren.	Drahtlose Verbindung
Seminarraum (in vitro Evaluation, ohne mögliche Störungen, Beobachtung durch Evaluator und zwei Kameras)	Studenten (20-30), Wien	Haptische Interaktion (Berühren, Kippen, Bewegen des Gerätes)	PDA mit Windows mobile	
		Hauptszenario: Anwendung starten, bestimmten Ort finden, diesen auskundschaften, den nächsten Ort finden, Anwendung ausschalten.		

Abbildung 18: Nutzungskontext SYM

5.2.3.3 *Beschreiben der Softwarearchitektur*

Klassendiagramme und der Quelltext der prototypischen Anwendung lagen vor. Zur Beschreibung der Softwarearchitektur wurden ein UML-Komponentendiagramm, CRC-Karten (Class-Responsibility-Collaboration) mit Erläuterungen ihrer Verantwortlichkeiten und Kollaborationen beschrieben sowie ein UML-Sequenzdiagramm erstellt.

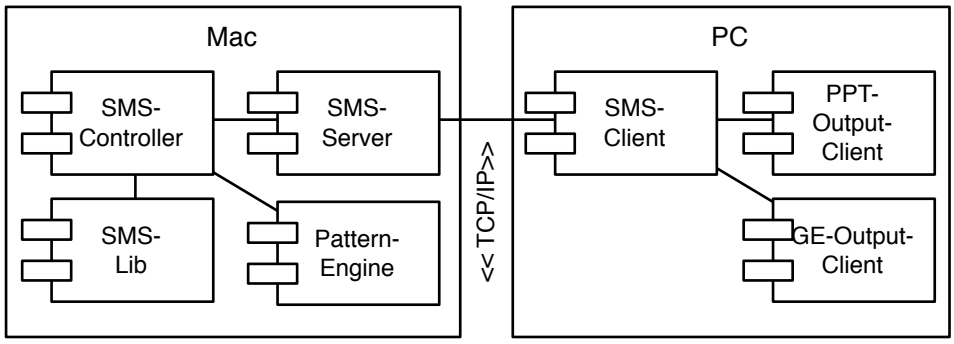

Abbildung 19: Komponentendiagramm von SYM

Das Komponentendiagramm in Abbildung 19 zeigt die Struktur der Anwendung. Auf dem Mac befinden sich die Komponenten `SMSController`, `SMSLib`, `PatternEngine` und `SMSServer`. SMS steht für SuddenMotionSensor, also den Beschleunigungsaufnehmer. `SMSLib` und `PatternEngine` sind mit dem `SMSController` verbunden, dieser kollaboriert mit dem `SMSServer`. Der `SMSServer` auf dem Mac und der `SMSClient` auf dem PC kommunizieren über TCP/IP miteinander. Auf dem PC kommuniziert die Komponente `SMSClient` mit den Komponenten `PPTOutputClient` und `GEOutputClient`.

SMSServer	
Sammelt Sensorenstatus oder Interaktionspatterns	SMS Controller
Transformiert Interaktionsmuster für PPTOutputClient und GEOutputClient	SMSClient
Protokoll: TCP/IP	

SMSClient	
Abfrage des Serverstatus Abfrage von Eingabedaten Abfrage der Position oder Interaktion Berechnen der Eingabedaten Verteilung von Events an PPTOutputClient und GEOutputClient	SMS-Server

SMSController	
Startet Server Setzt Intervalle für Sensor-Query	SMSServer
Startet Pattern Engine Evaluiert Eingabedaten Produziert Interaktionspatterns	Pattern-Engine
Speichert Events (Bewegungsinteratkions patterns) Speichert Sensorstatus (Position des Endgerätes)	SMSLib

PPTOutputClient/GEOutputClient	
Abfrage des Serverstatus Abfrage von Eingabedaten Abfrage der Position oder Interaktion Berechnen der Eingabedaten Anzeige der Interaktion auf Bildschirm	SMSClient

PatternEngine	
Analysiert Rohdaten Generiert Events und leitet sie an SMS-Controller	SMS-Controller

SMSLib	
Sammeln der Rohdaten der Beschleunigungs-aufnehmer	SMS-Controller

Abbildung 20: CRC-Karten von SYM

Die Class-Responsibility-Collaboration-Diagramme [BC89] in Abbildung 20 erläutern, wofür die Komponenten verantwortlich sind und mit welchen Komponenten sie kollaborieren.

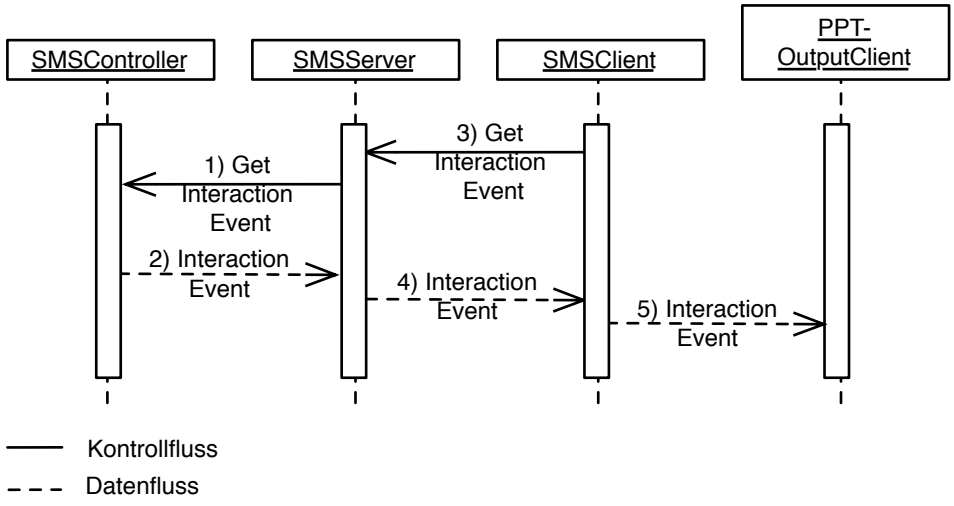

Abbildung 21: Sequenzdiagramm von SYM

Das Sequenzdiagramm in Abbildung 21 zeigt, wie die Hauptinteraktion abläuft. Der `SMSController` startet den `SMSServer` und setzt die Intervalle, in dem der Beschleunigungsabnehmer abgefragt wird. Er startet außerdem die Komponenten `SMSLib`, welche die Rohdaten der Beschleunigungsaufnehmer sammelt und `PatternEngine`, welche die Rohdaten analysiert, Events generiert und an den `SMSController` weiterleitet. Der `SMSController` evaluiert diese Eingabedaten und produziert Interaktionspatterns, die an den `SMSServer` weitergeleitet werden. Der `SMSServer` sammelt diese Daten und transformiert sie in Interaktions-Events für den `PPTOutputClient` (manchmal auch als `PPTClient` abgekürzt) und den `GEOutputClient` (manchmal auch als `GEClient` abgekürzt). Diese Daten werden über TCP/IP an den `SMSClient` versendet, der sich auf dem Mac befindet. Er fragt regelmäßig den Status des `SMSServers` sowie die Eingabedaten, die Position oder die Interaktion des Nutzers ab. Er berechnet daraus die Eingabedaten, die an die Komponenten `PPTOutputClient` und `GEOutputClient` geschickt werden. Diese beiden Komponenten fragen beim `SMSClient` den Status des `SMSServers` ab, ebenfalls die Eingabedaten, Position und Interaktion. Jede Komponente berechnet die Eingabedaten für die eigene Anwendung und zeigt die Interaktion auf dem Bildschirm an.

Der `SMSController` produziert Interaktions-Events, die auf regelmäßige Anfrage vom `SMSServer` zu diesem gesendet werden. Der `SMSServer` sendet sie auf regelmäßige Anfrage vom `SMSClient` an diesen. Von dort aus werden sie an den `PPTOutputClient` bzw. den `GEOutputClient` geschickt. `SMSServer` und `SMSClient` kommunizieren in einem regelmäßigen Request-Response-Zyklus über TCP/IP miteinander.

Der Google Earth Output Client `GEOutputClient` implementiert mit Hilfe des COM Interfaces von Google Earth die Verbindungen zu der Anwendung. Kontrolliert wird der Globus durch die Navigation in vier Richtungen, Kippen und Zoomen. Auf diese Navigationsaufgaben kann über Modi zugegriffen werden.

Der Powerpoint Output Client `PPTOutputClient` implementiert Verbindungen zur Anwendung Microsoft PowerPoint mit Hilfe des COM Interfaces von Powerpoint. Die Implementierung ermöglicht das Navigieren in einer Präsentation: vorwärts, rückwärts, Sprung zum Anfang, Sprung zum Ende. Wird das Gerät nach rechts geneigt oder wird auf oder neben das Gerät geklopft, interpretiert das Interaktionsgerät dies als „vorwärts navigieren“. Analog dazu bedeutet das Kippen nach links „zurück navigieren“. Nach vorn kippen bedeutet „Navigieren zum Ende“ und zurück bedeutet „Navigieren zum Anfang“.

Es soll möglich sein, dass das Interaktionsgerät von vielen Anwendungen auf verschiedenen Ausgabegeräten (wie dem PC) verwendet wird. Es soll außerdem möglich sein, verschiedene Interaktionsgeräte zu nutzen (analog zum Mac).

5.2.4 *Durchführung der Fallstudie SYM: SATURN-Phase 2*

5.2.4.1 *Mapping der Interaktionsszenarios zu Anforderungen*

Aus der ersten Phase ist bekannt, welche Usability-Anforderungen realisiert werden sollen. Aus CASSINI wurden zuerst folgende Interaktionsszenarios ausgewählt: Abbrechen (Cancel), System-Feedback und Mehrkanalzugang (Multi-Channel Access). Das letztgenannte Interaktionsszenario soll erweitert werden: die Anwendung soll mit verschiedenen Eingangsgeräten, aber auch mit verschiedenen Ausgabegeräten nutzbar sein. Außerdem soll die Verbindung zwischen den Geräten abgesichert und wiederhergestellt werden.

- o Absicherung und die Automatisierung des Verbindungsaufbaus nach einem Verbindungsverlust
- ✓ Abbrechen von Interaktionen
- ✓ gutes Feedback für die Benutzer
- ✓ Ergänzen neuer Endgeräte, damit Nutzer mit einem Endgerät ihrer Wahl mit der Anwendung interagieren können (Anwendungsfall für das Museum)

5.2.4.2 *Auswahl aus CASSINI*

Unabhängig voneinander wählten die Teilnehmer weitere Interaktionsszenarios aus: Wiederherstellung nach Betriebsausfall (Recovering from Failure), Mehrstufige Hilfe (Multi-Level Help), Gleiche Bedienung verschiedener Ansichten (Operating Consistently Across Views) und Konfliktfreie Nebenläufigkeit (Non-conflicting Application Concurrency).

5.2.4.3 *Bestimmen der zu evaluierenden Interaktionsszenarios*

Ausgewählt wurden die Interaktionsszenarios, die während des Walkthroughs ermittelt wurden: Abbrechen (Cancel), Add New Input/Output Device (abgeleitet vom Interaktionsszenario Mehrkanalzugang), System-Feedback, Safe the User's Work (abgeleitet vom Interaktionsszenario Wiederherstellung nach Betriebsausfall). Die anderen Interaktionsszenarios wurden für einen zukünftigen Durchgang festgehalten.

- ✓ Absicherung und die Automatisierung des Verbindungsaufbaus nach einem Verbindungsverlust – Interaktionsszenario System-Feedback, Safe the User's Work (abgeleitet vom Interaktionsszenario Wiederherstellung nach Betriebsausfall).
- ✓ Abbrechen von Interaktionen – Interaktionsszenario Abbrechen (Cancel)
- ✓ gutes Feedback für die Benutzer – Interaktionsszenario System-Feedback,
- ✓ Ergänzen neuer Endgeräte, damit Nutzer mit einem Endgerät ihrer Wahl mit der Anwendung interagieren können (Anwendungsfall für das Museum) – Interaktionsszenario Add New Input/Output Device (abgeleitet vom Interaktionsszenario Mehrkanalzugang)

Abbildung 22 zeigt, wie die Auswahl beim Durchführen der Methode dokumentiert wurde.

Nr.	Anforderung	Begründung	Szenario	Architekt	Analyst	Auswahl
	Intution					
1	Switch between Tasks	Kontext: Walkthrough	Feedback	siehe 8	siehe 8	siehe 8
2	Consistency in style and dialog sequences	Katalog	Operating Consistently across Views	die Interaktion bei den verschiedenen Anwendungen muss konsistent sein	Zustimmung, aber Wunsch, erst einmal die Funktionalität zu prüfen	später
3	Consistent work at different views	Kontext	Operating Consistently across Views	siehe 2	siehe 2	siehe 2
4	Special guidance to new users	Katalog	Multi-Level Help, Provide Context Related Help	prüfen, ob Hilfe angeboten wird	nicht ausgewählt	später
5	Work at the User's Pace	Katalog	User' Pace	nicht ausgewählt	nicht ausgewählt	nicht ausgewählt
6	System Feedback	Kontext: Walkthrough	Feedback	ein Problem ist wahrscheinlich	Anforderung wichtig basierend auf Kontext (Hauptziel des Systems ist die Interaktion)	erster Durchgang
	Robustheit					
7	Checking for Correctness	Kontext	Checking for Correctness			
8	Cancel Functionality	Kontext: Walkthrough	Abbrechen (Cancel)	Stoppen der Interaktion muss möglich sein.	Stimmt zu.	erster Durchgang
9	Safe the User's Work	Katalog	Recovering from Failure	es könnte ein Problem geben	Netzwerkverbindungen sind instabil	erster Durchgang
10	Prevent Resource Conflicts	Katalog	Non-conflicting Device Usage	Wichtig bei Erweiterung	es könnte ein Problem geben, ist wichtig	später
	Flexibilität					
11	Multichannel-Access	Kontextfaktoren, Katalog	Multi-channel-Access	spezialisiert als Add New Input/Output Device	siehe 12	siehe 12
12	Add New Input/Output Devices	Kontextfaktoren, Katalog	Add New Input/Output Device	Anforderung ist Ergänzen von I/O	Anforderung wichtig basierend auf Kontext (Wesentliches Merkmal der Anwendung)	erster Durchgang

Abbildung 22: SYM – Auswahl der Interaktionsszenarios

Danach wurde in der dritten SATURN-Phase ein Interaktionsszenario nach dem anderen mit den folgenden Evaluationsformularen untersucht.

5.2.5 *Durchführung der Fallstudie SYM: SATURN-Phase 3*

5.2.5.1 *Canceling Commands (Abbrechen, Cancel)*

Abbrechen (Cancel)

Allgemeine Informationen Nummer: 4 Kurzname: Abbrechen (Cancel) Anforderungskategorie: Robustheit Interaktionskategorie: Executing, Repeating and Revoking Commands	**In die Analyse aufgenommen via** Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Begründung der Auswahl** Architekt: Stoppen der Interaktion muss möglich sein. Analyst: Stimmt zu.
Szenario Umwelt: Museum Initiator: „Users“ Stimulus: „started an operation but now don't want it to be executed any longer“ Umgebung: Mac/PC Response: „The system immediately stops execution as users activate the cancelling option.“ Response Measure: „Cancellation is performed and communicated to the user within a certain time span (A) System state before starting the operation is restored. (A)“	**Rationale, Referenzen** Usability-Attribute: Effectiveness, Efficiency, Safety, Learnability HCI-Rational: Kontrolle über das System, Benutzungs- oder Systemfehler ohne große Auswirkungen HCI-Patterns: *Canceling Commands* [BJK01], *Cancelability* [Tid06] SE-Rational: An operation cannot cancel itself. There have to be made monitoring and executing provisions to achieve a robust cancel interaction. SE-Patterns: [Fol05] *Cancel*: Provision for the componetns monitoring user input, this should be independent, concurrent; components processing actions can be interrupted, consequences of actions are rolled back. Projektreferenzen: bisher keine.

Analyse
Use Case(s): Stoppen einer Navigation (bei beiden Anwendungen)

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
1	Cancel	Navigieren (Hauptinteraktion)	`SMSServer`, `SMSClient`,	TCP/IP	keins	Existenz einer spezialisierten Distributionskomponente
2	Cancel	Navigieren (Hauptinteraktion)	`SMSServer`, `SMSClient`,	Interface für AnwendungsController	keins	Interaktions-Events werden durch die Anwendungs-Controller transformiert
3	Cancel	Navigieren (Hauptinteraktion)	`SMSServer`, `SMSClient`, `GEClient`, `PPTClient`	TCP/IP, Interface für Anwendung-Controller	Canceling Commands [BJK01]	Behandlung der Interaktion durch die Anwendungs-Controller `PPTOutputClient` und `GEOutputClient`: Controller für Google Earth erlaubt wegen Queue keine direkte Manipulation

Sensitive Entscheidungen: 1 (gut), 2 (gut), 3 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
1	Existenz einer spezialisierten Distributionskomponente	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
2	Interaktions-Events werden durch die Anwendungs-Controller transformiert	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
3	Behandlung der Interaktion durch die Anwendungs-Controller `PPTOutputClient` und `GEOutputClient`	Controller erlaubt keine direkte Manipulation, Interaktion wird ggf. nicht abgebrochen → Response wird nicht erreicht	2	Änderung des Interfaces und/oder des Protokolls	1 oder 2	[-] Performanz	ja

Zusammenfassung
Usability-Problem: Zeitverzögerung beim Prototypen für die Interaktion mit Google Earth. Wesentliches geht nicht: die Benutzer sehen nicht, welche Aktion das System durchführt. Das System stoppt nicht, d.h. die Response wird nicht erreicht. → 2

Abbildung 23: Evaluationsformular Cancel, Seite 1

SA-Sensitivität: Die Abarbeitung der Interaktion bei Google Earth sind zu ändern. Eine Alternative ist ein anderes Kommunikationsprotokoll (UDP statt TCP/IP) → Änderungsaufwand ist 1 oder 2.

Tradeoffs: [-] Performanz (der Einsatz einer FIFO-Queue führt zu Verzögerungen), [+] Modifizierbarkeit (durch Kapselung)
Andere(s) Szenario(s): Feedback (der Einsatz der FIFO-Queue führt zu Verzögerungen)
Architekturdiagramm(e): Sequenzdiagramm Feedback, Klassendiagramm InterfaceApplicationController

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,1) oder (2,2)
Ziele: ASL 1. Iteration: (0,0) – höchste Priorität ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Beobachten und bewerten, wie Benutzer die Verzögerung bei Google Earth wahrnehmen.

Sonstige Kommentare: keine.

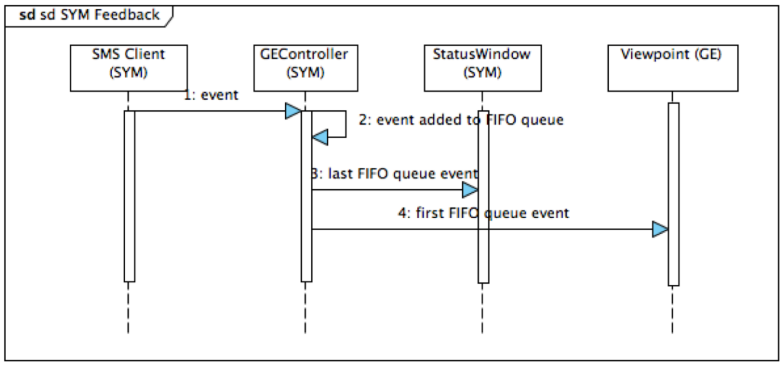

Abbildung: SD SYM Feedback

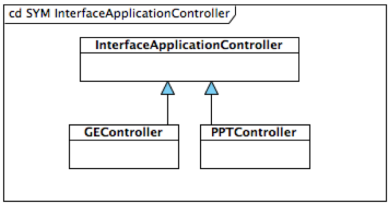

Abbildung: CD SYM `InterfaceApplicationController`

Abbildung 24: Evaluationsformular Cancel, Seite 2

5.2.5.2 *Add New Input/Output Device*

Add New Input/Output Device

Allgemeine Informationen	**In die Analyse aufgenommen via (zutreffendes unterstreichen)**
Nummer: 21 Kurzname: **Add New Input/Output Device** (basiert auf Multi-Channel Access) Anforderungskategorie: Flexibilität Interaktionskategorie: Exchange/Cooperation Within and Between Systems	Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Rating mit Begründung** Architekt: Anforderung ist Ergänzen von I/O Analyst: Anforderung wichtig basierend auf Kontext (Wesentliches Merkmal der Anwendung)

Szenario	**Rationale, Referenzen**
Umwelt: Initiator: „Users“ (Administrator) Stimulus: „want to access the system using a specific type of i/o device.“ Umgebung: Mobiles Endgerät oder Mac und Mac und/oder PC Response: „The system allows for connections with different types of i/o devices. (A)“ Respones Measure: „The users' I/O device is connected to the computer hardware and works properly with the application.“	Usability-Attribute: Effectiveness, Utility Usability-Anforderung: Rahmenbedingung mobiler Anwendungen: techn. Fortschritt bzgl. I/O-Technologie HCI-Rational: Verwenden einer Anwendung mit einem gewünschtem Eingabegerät HCI-Patterns: (keine Angabe) SE-Rational: There has to be a provision that different i/o devices are identified and provided with compatible data. SE-Patterns: Multi Channel Access [Fol05] Projektreferenzen: keine

Analyse

Use Case(s): Ergänzung weiterer Controller für Word und Firefox; Ergänzung eines weiteren Sudden Motion Sensors in einem PDA mit drahtloser Verbindung über Bluetooth

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
4	Multi-Channel Access	Ergänzung SMS im PDA	`SMSServer`, `SMSController`,`SMSLibrary`, `PatternEngine`	neue Schnittstellen	keins	Nicht-Existenz einer Regelung zum Ergänzen weiterer Eingabegeräte
5	Multi-Channel Access	Ergänzung weiterer Controller	`SMSClient`, Client für Word/ Firefox	`Interface-Application-Controller`	keins	Nicht-Existenz einer Regelung zum Ergänzen weiterer Ausgabegeräte

Sensitive Entscheidungen: 4 (kritisch), 5 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
4	Nicht-Existenz einer Regelung zum Ergänzen weiterer Eingabegeräte	Benutzer können keine weiteren Eingabegeräte ergänzen > Response nicht realisiert	2	Neue Schnittstellen und neue Controller	2	[+] Modifizierbarkeit [?] Performanz	ja
5	Nicht-Existenz einer Regelung zum Ergänzen weiterer Ausgabegeräte	Benutzer können das System nicht mit weiteren Anwendungen oder anderen Geräte verwenden > Response nicht realisiert	2	Änderung des Interfaces und/oder des Protokolls, verschiedene Architekturen möglich (Frage nach gleichzeitiger Bedienung mehrerer Clients)	3	[+] Modifizierbarkeit [?] Performanz	ja

Abbildung 25: Evaluationsformular Add New Input/Output Device, Seite 1

Zusammenfassung
Usability-Problem: Weder Eingabe- noch Ausgabegeräte sind ohne Weiteres integrierbar. Das Szenario ist also gar nicht umgesetzt > 2

SA-Sensitivität: Erweiterungen erfordern unbekannte Änderungen – es sind verschiedene Alternativen möglich > 3

Tradeoffs: [+] Modifizierbarkeit (durch Kapselung), Performanz wird beeinflusst werden, aber wie, ist eine offene Frage.
Andere(s) Szenario(s): Feedback (hinsichtlich der neuen Eingabe- und Ausgabegeräte)
Architekturdiagramm(e): Verteilungsdiagramme SYM und Alternatives SYM

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,3)
Ziele: ASL 1. Iteration: (2, 1 oder 2) – Klarheit über Aufwand, ASL 2. Iteration: (0,0)

Usability-Evaluation: Test, ob und wie einfach Administratoren I/O-Geräte anschließen können.

Sonstige Kommentare:
- Vorgesehen, dass die Eingabe modularisiert wird und dass jede Art von Bewegungsdaten aufgenommen und interpretiert werden kann;
- außerdem Ziel, dass Benutzer zwischen den Programmen wechseln können

Abbildung 26: Evaluationsformular Add New Input/Output Device, Seite 2

5.2.5.3 *Observing System State (System Feedback)*

Feedback

Allgemeine Informationen	**In die Analyse aufgenommen via (zutreffendes unterstreichen)**
Nummer: 1 Kurzname: Feedback Anforderungskategorie: Intention	Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes
Interaktionskategorie(n): Orientation, Structuring and Displaying Content/Information/Data, Adaptation (by Users/to Users/for Tasks)	**Rating mit Begründung** Architekt: ein Problem ist wahrscheinlich Analyst: Anforderung wichtig basierend auf Kontext (Hauptziel des Systems ist die Interaktion)
Szenario Umwelt: Museum Initiator: „Users“ (Administrator) Stimulus: „want to stay informed about the system's state, activities and changes.“ Umgebung: Mobiles Endgerät oder Mac und Mac und/oder PC Response: „The system picks the data required by the user and presents them according to human needs and capabilities. “ Response Measure: „Users are supplied with all necessary data.U Users perceive all necessary data. (U) The data perceived by the users are up to date. (U)“	**Rationale, Referenzen** Usability-Attribute: Efficiency, Safety, Utility, Learnability Usability-Anforderung: Benutzer müssen eine Reaktion auf ihre Aktion bemerken und wissen, was das System gerade macht HCI-Patterns: keine. SE-Rational: Observing System State [BJK01], System Feedback [Fol05] Projektreferenzen: keine.

Analyse
Use Case(s): Interaktion mit Google Earth (GE), Interaktion mit Microsoft PowerPoint (PPT).

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
6	Feedback	Interaktion mit Google Earth	`SMSClient`, `GEController`, `StatusWindow`, `Viewpoint`	Interface für die Anwendungs-Controller	keins	GUI des Controllers zeigt die soeben gestartete Interaktion an (als Systemstatus des `SMSControllers`)
7	siehe 6.	siehe 6.	siehe 6.	siehe 6.	siehe 6.	GUI des Controllers zeigt aber nicht die momentan durchgeführte, bearbeitete, Interaktion an
8	siehe 6.	siehe 6.	siehe 6.	siehe 6.	siehe 6.	Verwendung von TCP/IP als Netzwerkprotokoll

Sensitive Entscheidungen: 6 (nein), 7 (ja), 8 (ja)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
6	GUI des Controllers zeigt die soeben gestartete Interaktion an	Die GUI zeigt die von Benutzern gestartete Interaktion an.	0	keine Änderung nötig	0	keine	nein
7	GUI des Controllers zeigt aber nicht die momentan durchgeführte, bearbeitete, Interaktion an	Die GUI zeigt aber nicht an, was das System gerade bearbeitet, sondern nur die vom Benutzer durchgeführte Interaktion. Es muss aber die momentan durchgeführte Interaktion muss angezeigt werden, damit die Benutzer nicht dazu verleitet werden, eine weitere Interaktion zu starten. → Response nicht realisiert	2	Komponenten der OutputClients, Anwendungs-Controller anpassen	1	keine	nein
8	Verwendung von TCP/IP als Netzwerkprotokoll	Bei der Verwendung des TCP/IP Protokolls entsteht ein hoher Kommunikationsaufwand durch die Anfra-	2	Einsatz von TCP/IP und UDP	2	[+] Performanz [+] Modifizierbarkeit	ja

Abbildung 27: Evaluationsformular Feedback, Seite 1

gen.

Zusammenfassung
Usability-Problem: Szenario wird nur teilweise umgesetzt, das wesentliche – der Status des Systems – aber nicht → 2

SA-Sensitivität: Status von GE muss abgefragt werden, jeweils eine Funktion ist in Controllern zu ergänzen, das Interface entsprechend. Die Verwendung des TCP/IP-Protokolls muss überdacht werden. Alternativen sind UDP oder eine Kombination von TCP/IP und UDP. → 2

Tradeoffs: [-] Performanz (der Einsatz einer FIFO-Queue führt zu Verzögerungen), [+] Modifizierbarkeit (durch Kapselung)
Andere(s) Szenario(s): Feedback
Architekturdiagramm(e): Sequenzdiagramm Feedback

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,2)
Ziele: ASL 1. Iteration: (0,0) – höchste Priorität, ASL 2. Iteration: (0,0)

Usability-Evaluation: Response Measure testen „Users are supplied with all necessary data. (U) Users perceive all necessary data. (U) The data perceived by the users are up to date. (U)"

Sonstige Kommentare:
keine

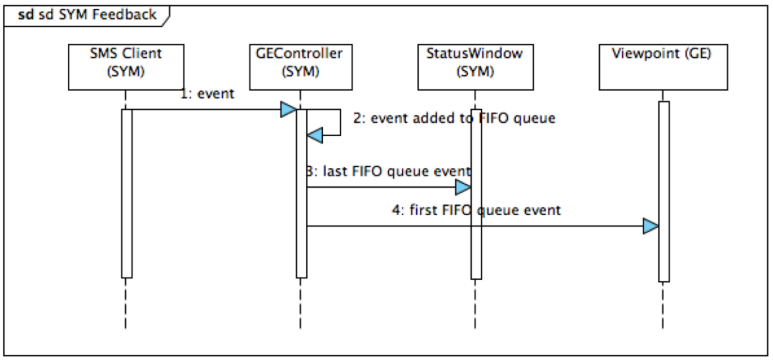

Abbildung: SD SYM Feedback

Abbildung 28: Evaluationsformular Feedback, Seite 2

5.2.5.4 *Recovering from Failure*

Recovering from Failure

Allgemeine Informationen
Nummer: 28
Kurzname: Recovering from Failure
Anforderungskategorie: Robustheit

Interaktionskategorie(n): Error Handling and Help

In die Analyse aufgenommen via (zutreffendes unterstreichen)
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Rating mit Begründung
Architekt: es könnte ein Problem geben
Analyst: Netzwerkverbindungen sind instabil

Szenario
Umwelt: Museum
Initiator: „System“
Stimulus: „does not work properly or crashes“
Umgebung: Mobiles Endgerät oder Mac und Mac und/oder PC
Response: „The system minimizes (or provides the users with the means to reduce) the amount of work lost due to the failure.“
Qualitätsziele: „The users' work is preserved (to a certain degree). (A)“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Efficiency, Safety
Usability-Anforderung: Rahmenbedingung mobiler Anwendungen: Netzwerkabhängigkeit
HCI-Rational: „A system may suddenly stop functioning while a user is working. Users should be provided
with the means to reduce the amount of work lost from system failures.“
HCI-Patterns: keins
SE-Rational: Recovering from Failure [BJK01]
Projektreferenzen: keine.

Analyse
Use Case(s): Client verliert die Verbindung

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
9	Recovering from Failure	Client verliert Verbindung	`SMSServer`, `SMSClient`	TCP/IP	keins	Nicht-Existenz einer Vorrichtung zum Monitoring bzw. dem Management der Konnektivität (in beide Richtungen: Eingabe und Ausgabe)

Sensitive Entscheidungen: 9 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
9	Nicht-Existenz einer Vorrichtung zum Monitoring bzw. dem Management der Konnektivität (in beide Richtungen: Eingabe und Ausgabe)	Der Verlust der Netzwerkkonnektivität führt dazu, dass die Anwendungen stoppen. Die Navigationsdaten (aktuelle Sicht) gehen verloren. Response nicht realisiert.	2	Eine weitere Komponente zur Netzwerküberwachung und automatische Verbindung ist notwendig, Anpassung der Anwendungs-Controller. Allerdings sind sowohl Eingabe als auch Ausgabe betroffen.	2	[-] Fehlertoleranz	ja

Zusammenfassung
Usability-Problem: Bei einem Netzausfall ist das System nicht mehr verwendbar. Nach erneuter Verbindung können die Benutzer nicht an der gleichen Stelle weitermachen wie vor dem Fehler, die Nutzerdaten gingen also verloren. Szenario wird also nicht umgesetzt → 2

SA-Sensitivität: Eine neue Komponente muss ergänzt und integriert werden. Damit ergeben sich Modifikationen in den dann beteiligten Komponenten. Es sind aber sowohl die Eingabe als auch die Ausgabe betroffen. → 2

Tradeoffs: [+] Fehlertoleranz gegenüber Netzwerkverlust
Andere(s) Szenario(s): keine
Architekturdiagramm(e): Verteilungsdiagramm SYM

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,2)
Ziele: ASL 1. Iteration: (0,0) – höchste Priorität, ASL 2. Iteration: (0,0)

Usability-Evaluation: der Response-Measure: „Users are supplied with all necessary data. (U) Users perceive all necessary data. (U) The data perceived by the users are up to date. (U)“

Abbildung 29: Evaluationsformular Recovering from Failure, Seite 1

Sonstige Kommentare:
- Kann der Status programmintern zwischengespeichert werden? Wie definiert und aktualisiert man den Status, wenn nicht?
- beim Environment im Museum muss der SMS-Controller selbständig starten

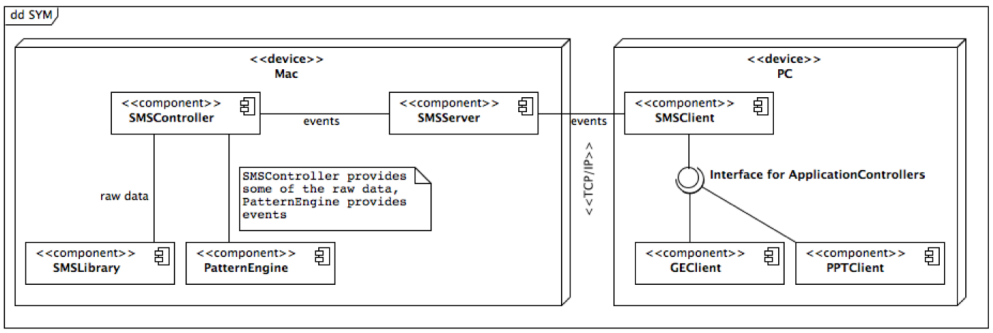

Abbildung: Verteilungsdiagramm SYM

Sonstiges:

- Es wird eine GUI-Komponente für Admin des SMS-Controllers benötigt, mit Benutzereingaben über Maus/Tastatur am Mac (MVC)

Abbildung 30: Evaluationsformular Recovering from Failure, Seite 2

5.2.6 *Durchführung der Fallstudie SYM: SATURN-Phase 4*

5.2.6.1 *Architektur-Support-Level*

Szenarios	Begründung	Sensitive Entscheidungen	Unterstützung des Szenarios	Max. Einfluss auf Usability, Softwarearchitektur (ASL aktuell)	ASL0 (aktuell)	ASL1 (folgende Iteration)	ASL 2 (folgende Iteration)
Abbrechen (Cancel)	Kontext: Walkthrough	S1, S2, S3 (k)	nein	2,1	2,1	0,0	0,0
Add New Input/Output Device	Kontextfaktoren, Katalog	S4 (k), S5 (k), S6	nein	2,3	2,3	0,1 oder 2	0,0
Feedback	Kontext: Walkthrough	S7, S8 (k)	nein	2,2	2,2	0,0	0,0
Recovering from Failure	Katalog	S9 (k)	nein	2,2	2,2	2,2	0,0

Einfluss auf Usability	Einfluss auf Softwarearchitektur
0... Die Anforderung wird umgesetzt.	0... Keine Modifikationen werden erwartet.
1... Die Anforderung wird teilweise umgesetzt.	1... Einfache Modifikationen werden erwartet.
2... Die Anforderung wird nicht umgesetzt.	2... Komplexe Modifikationen werden erwartet.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.	3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

Abbildung 31: Tabelle Architektur-Support-Level „Shake Your Mac“

Die Tabelle 31 fasst die Bewertung der Interaktionsszenarios zusammen: es wurden vier Interaktionsszenarios untersucht, drei davon wurden in Phase 1, eins in Phase 2 ermittelt. Das ASL für diese Version der Architektur ist niedrig, da keines der vier untersuchten Interaktionsszenarios hypothetisch durchgeführt werden kann.

Es wurden Modifikationen vorgeschlagen: zwei einfache (Interaktionsszenarios Abbrechen und Feedback), eine komplexe (Recovering from Failure), eine mit unbekanntem Aufwand (Add New Input/Output Devices).

Basierend auf den Zielvorgaben, die bei der Evaluation der Interaktionsszenarios nach Dringlichkeit vergeben wurden, wird der Architekt zuerst die Interaktionsszenarios Abbrechen und Feedback in seinem Entwurf umsetzen, dann soll die Architektur erneut (von einer anderen Person) überprüft werden. Anschließend sollen die beiden anderen Interaktionsszenarios Add New Input/Output Device und Recovering from Failure integriert werden, denn sie erfordern umfassendere Modifikationen.

5.2.6.2 *Architekturentscheidungen auflisten*

Neun szenario-beeinflussende Architekturentscheidungen wurden ermittelt. Die Tabellen 32 und 33 bieten einen Überblick.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Abbrechen (Cancel)	ASL 1	1	Existenz einer spezialisierten Distributionskomponente	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
			2	Interaktions-Events werden durch die Anwendungs-Controller transformiert	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
			3	Behandlung der Interaktion durch die Anwendungs-Controller `PPTOutputClient` und `GEOutputClient`	Controller erlaubt keine direkte Manipulation, Interaktion wird ggf. nicht abgebrochen → Response wird nicht erreicht	2	Änderung des Interfaces und/oder des Protokolls	1 oder 2	[-] Performanz	ja
Flexibilität	Add New Input/Output Device	ASL 2	4	Nicht-Existenz einer Regelung zum Ergänzen weiterer Eingabegeräte mit entsprechenden	Benutzer können keine weiteren Eingabegeräte ergänzen → Response nicht realisiert	2	Neue Schnittstellen und neue Controller	3	[+] Modifizierbarkeit [?] Performanz	ja
			5	Nicht-Existenz einer Regelung zum Ergänzen weiterer Ausgabegeräte mit entsprechenden	Benutzer können das System nicht mit weiteren Anwendungen oder anderen Geräte verwenden → Response nicht realisiert	2	Änderung des Interfaces und/oder des Protokolls, verschiedene Architekturen möglich (Frage nach gleichzeitiger Bedienung mehrerer Clients)	3	[+] Modifizierbarkeit [?] Performanz	ja
Intuition	Feedback	ASL 1	6	GUI des Controllers zeigt die soeben gestartete Interaktion an	Die GUI zeigt die von Benutzern gestartete Interaktion an.	0	keine Änderung nötig	0	keine	nein
			7	GUI des Controllers zeigt aber nicht die momentan durchgeführte, bearbeitete, Interaktion an	Die GUI zeigt aber nicht an, was das System gerade bearbeitet, sondern nur die vom Benutzer durchgeführte Interaktion. Es muss aber die momentan durchgeführte Interaktion muss angezeigt werden, damit die Benutzer nicht dazu verleitet werden, eine weitere Interaktion zu starten. → Response nicht realisiert	2	Komponenten der OutputClients, Anwendungs-Controller anpassen	1	keine	nein
			8	Verwendung von TCP/IP als Netzwerkprotokoll	Bei der Verwendung des TCP/IP Protokolls entsteht ein hoher Kommunikationsaufwand durch die Anfragen.	2	Einsatz von TCP/IP und UDP	2	[+] Performanz [+] Modifizierbarkeit	ja
Robustheit	Recovering from	ASL 2	9	Nicht-Existenz einer Vorrichtung zum	Der Verlust der Netzwerkkonnektivi-	2	Eine weitere Komponente zur	2	[-] Fehlertoleranz	ja

Abbildung 32: SYM-Architekturentscheidungen, Seite 1

	Failure	Monitoring bzw. dem Management der Konnektivität (in beide Richtungen: Eingabe und Ausgabe)	tät führt dazu, dass die Anwendungen stoppen. Die Navigationsdaten (aktuelle Sicht) gehen verloren. Respons nicht realisiert.	Netzwerküberwachung und automatische Verbindung ist notwendig, Anpassung der Anwendungs-Controller. Allerdings sind sowohl Eingabe als auch Ausgabe betroffen. Aufwand derzeit nicht gut einschätzbar.

Einfluss auf Usability	Einfluss auf Softwarearchitektur
0... Die Anforderung wird umgesetzt.	0... Keine Modifikationen werden erwartet.
1... Die Anforderung wird teilweise umgesetzt.	1... Einfache Modifikationen werden erwartet.
2... Die Anforderung wird nicht umgesetzt.	2... Komplexe Modifikationen werden erwartet.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.	3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

Abbildung 33: SYM-Architekturentscheidungen, Seite 2

5.2.6.3 *Themen beschreiben*

KOMMUNIKATION Alle Architekturentscheidungen zum Thema Kommunikation sind in Tabelle 34 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Abbrechen (Cancel)	ASL 1	1	Existenz einer spezialisierten Distributionskomponente	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
Robustheit	Abbrechen (Cancel)	ASL 1	2	Interaktions-Events werden durch die Anwendungs-Controller transformiert	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine. Kapselung sinnvoll.	0	[+] Performanz [+] Modifizierbarkeit	nein
Intuition	Feedback	ASL 1	8	Verwendung von TCP/IP als Netzwerkprotokoll	Bei der Verwendung des TCP/IP Protokolls entsteht ein hoher Kommunikationsaufwand durch die Anfragen.	2	Einsatz von TCP/IP und UDP	2	[-] Performanz	ja
Robustheit	Recovering from Failure	ASL 2	9	Nicht-Existenz einer Vorrichtung zum Monitoring bzw. dem Management der Konnektivität (in beide Richtungen: Eingabe und Ausgabe)	Der Verlust der Netzwerkkonnektivität führt dazu, dass die Anwendungen stoppen. Die Navigationsdaten (aktuelle Sicht) gehen verloren. Respons nicht realisiert.	2	Eine weitere Komponente zur Netzwerküberwachung und automatische Verbindung ist notwendig, Anpassung der Anwendungs-Controller. Allerdings sind sowohl Eingabe als auch Ausgabe betroffen. Aufwand derzeit nicht gut einschätzbar.	2	[-] Fehlertoleranz	ja

Abbildung 34: S1, S2, S8, S9 betreffen Kommunikation

Erläuterung: Die Komponenten `SMSServer` und `SMSClient` kommunizieren über das Protokoll TCP/IP miteinander (S8), wobei nicht überwacht wird, ob die Komponenten noch miteinander verbunden sind (S9).

Eine Sub-Komponente innerhalb der Komponente `SMSClient` verantwortet die Aufgabe, Bewegungsdaten an die Komponenten `PPTOutputClient` und `GEOutputClient` zu senden (S1). Diese transformieren sie in spezifische Interaktions-Events ihrer Anwendung (S2).

Voraussichtliche Unterstützung für die Usability:

(+) Separation von der Behandlung allgemeiner Events (`SMSClient`) und ihrer Transformation in spezifische Interaktions-Events (`PPTOutputClient`, `GEOutputClient`) unterstützt Hauptinteraktion: Die Architekturentscheidungen S1 und S2 ermöglichen, dass erstens Interaktions-Events für jeden Client aufbereitet und damit voraussichtlich auch abgearbeitet werden können und zweitens, dass nötige Modifikationen durch beispielsweise die Aktualisierung eines Programmes erleichtert werden.

Hinweise auf mögliche Probleme bei der Benutzung:

(-) Kommunikationsprotokoll TCP/IP verlangsamt das Antwortverhalten: Die Verwendung des TCP/IP-Protokolls (S8) führt voraussichtlich wegen der in Intervallen abgefragten Events zu spürbaren Verzögerungen des Antwortverhaltens der Anwendung. Damit ist die Reaktion der Anwendung auf die Nutzerinteraktionen nicht unmittelbar.

(-) Ohne die automatische Überwachung und Wiederherstellung von Verbindungen ist die Benutzung nach einem Verbindungsabbruch unmöglich: Wegen S9 ist bei einem Verbindungsabbruch ein Administrator nötig, um die Verbindung neu zu erstellen. Das bedeutet, dass die Anwendung bei einem Verbindungsfehler nicht mehr benutzt werden kann.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(+) Modifizierbarkeit: Durch S2 und S3 können weitere Ausgabe-Clients integriert werden.

(-) Responsivität: Die Verwendung des TCP/IP-Protokolls (S8) kann wegen der in Intervallen abgefragten Events zu spürbaren Verzögerungen führen.

(-) Fehlertoleranz: Netzverbindungsabbrüche führen derzeit zum Systemausfall (S9).

Mögliche Modifikationen der Architektur:

- Ändern des Kommunikationsprotokolls ermöglicht Interaktion ohne Delay: Eine Alternative des Einsatzes von TCP/IP ist UDP. Dann werden Events weitergeleitet, sobald sie erstellt werden. Somit werden die Nutzer zumindest nicht die Verzögerung feststellen können, die durch das regelmäßige gesteuerte Abrufen geschieht. TCP/IP sollte aber weiterhin die notwendige bidirektionale Kommunikation beim Setup der Verbindung gewährleisten. Dies erfordert also die Anpassung von mehreren Komponenten und ihren Schnittstellen (Änderungskomplexität 2).
- Komponenten fürs Monitoring sollen es ermöglichen, einen Verbindungsverlust zu erkennen, zu behandeln und die Verbindung wiederherzustellen: Neue Komponenten für das Monitoring könnten die Verbindungen zwischen allen drahtlos verbundenen Komponenten der Anwendung überwachen und steuern. Insbesondere sollten Verbindungsabbrüche durch Fehler zügig automatisch behoben werden, damit die Nutzer keine oder nur eine kurze Unterbrechung erfahren. Dies erfordert also die Anpassung von mehreren Komponenten (Änderungskomplexität 2).

ANTWORTVERHALTEN Alle Architekturentscheidungen zum Thema Antwortverhalten sind in Tabelle 35 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Abbrechen (Cancel)	ASL 1	3	Behandlung der Interaktion durch die Anwendungs-Controller `PPTOutputClient` und `GEOutputClient`	Controller erlaubt keine direkte Manipulation, Interaktion wird ggf. nicht abgebrochen → Response wird nicht erreicht	2	Änderung des Interfaces und/oder des Protokolls	1 oder 2	[-] Performanz	ja
Intuition	Feedback	ASL 1	6	GUI des Controllers zeigt die soeben gestartete Interaktion an	Die GUI zeigt die von Benutzern gestartete Interaktion an.	0	keine Änderung nötig	0	keine	nein
			7	GUI des Controllers zeigt aber nicht die momentan durchgeführte, bearbeitete, Interaktion an	Die GUI zeigt aber nicht an, was das System gerade bearbeitet, sondern nur die vom Benutzer durchgeführte Interaktion. Es muss aber die momentan durchgeführte Interaktion muss angezeigt werden, damit die Benutzer nicht dazu verleitet werden, eine weitere Interaktion zu starten. → Response nicht realisiert	2	Komponenten der OutputClients, Anwendungs-Controller anpassen	1	keine	nein

Abbildung 35: Entscheidungen S3, S6, S7 betreffen das Thema Antwortverhalten

Erläuterung: Bei der Analyse wurde ermittelt, dass eine FIFO-Queue eine Nutzeraktion nach der anderen abarbeitet; es ist zwar ersichtlich, welche Interaktion der Client gerade abarbeitet, aber nicht, dass dies eine frühere Interaktion ist und dass ggf. noch weitere folgen (S7). Gleichzeitig gibt das User Interface dem Nutzer über ein Icon eine Rückmeldung dazu, welche Interaktion soeben vom Endgerät erkannt wurde (S6). Der Controller für Google Earth `GEOutputClient` erlaubt keine direkte Manipulation, d.h. es gibt keinen Event, der sofort ausgeführt wird, weshalb auch Abbrechen nicht funktionieren kann (S3).

Hinweise auf mögliche Probleme bei der Benutzung:

(-) Die UI zeigt an, welche Interaktion erkannt wurde, aber nicht, welche Interaktion von der Anwendung gerade ausgeführt wird. Das wirkt voraussichtlich verwirrend auf den Benutzer.

(-) Die Komponente `GEOutputClient` hält die Events in einer FIFO-Queue vor, sodass sie nacheinander an die Anwendung Google Earth weitergeleitet werden. Dies verhindert eine direkte Interaktion. Will der Nutzer wegen eines Versehens eine Interaktion abbrechen, ist das nicht möglich, denn eine FIFO-Queue erlaubt keine priorisierten Events, die vorn angestellt werden könnten. Im Gegenteil: die Interaktion, die er zum Abbrechen durchführte (beim Google Earth Client vielleicht eine Gegenbewegung), wird genauso abgearbeitet wie alle vorhergehenden. Da die Anwendung also einen Event nach dem anderen ausführt, wirkt es nach einer Weile so, als würde der Nutzer keine Kontrolle mehr haben, so dass ein Abbruch der Nutzung sehr wahrscheinlich ist.

Wechselwirkungen zwischen Usability und folgenden Qualitätsmerkmalen:

(-) Effizienz: Durch S3 und S7 werden Mehrfacheingaben provoziert.

Mögliche Modifikationen:

- Neue Interaktion für Cancel und neue Queue soll das Abbrechen ermöglichen: Es sollte ein Cancel-Event definiert werden, der nicht in die FIFO-Warteschlange aufgenommen, sondern direkt weitergeleitet wird (beispielsweise über priorisierte Events). Diese Lösung benötigt eine andere Art von Queue. Dies erfordert eine Änderung innerhalb der Komponente `GEOutputClient` sowie ein neues Interaktionsmuster, welches in den Komponenten `SMSLibrary` und `PatternEngine` integriert werden muss (Änderungskomplexität 1).

- Signalisierung im User Interface als Feedback für Nutzer, was das System gerade macht: Das User Interface sollte anzeigen, wenn das System die Queue abarbeitet und zeigen, dass die Nutzer mit der nächsten Interaktion warten sollen (z.B. durch eine kleine Animation wie eine Uhr, ein sich drehendes Rad). Dazu wird ein zweites Statusfenster notwendig, welches die gerade bearbeitete Interaktion des Programmes anzeigt. Dies erfordert nicht sehr komplexe Änderungen innerhalb der Komponenten `GEOutputClient` und `PPTOutputClient` (Änderungskomplexität 1).

- Alle Controller anpassen, um Erweiterbarkeit abzusichern: Um weitere Probleme dieser Art zukünftig auszuschließen, ist es sinnvoll, alle OutputClients zu ändern. In diesem Fall ist es notwendig, die Schnittstelle `InterfaceApplicationController` und alle davon erbenden OuputClients (`GEOutputClient` und `PPTOutputClient`) anzupassen. Diese Änderung betrifft verschiedene Komponenten, ist also aufwendig, aber nicht sehr komplex und innerhalb der aktuellen Struktur der Architektur (Änderungskomplexität 1).

ERWEITERBARKEIT Alle Architekturentscheidungen zum Thema Erweiterbarkeit sind in Tabelle 36 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Flexibilität	Add New Input/Output Device	ASL 2	4	Nicht-Existenz einer Regelung zum Ergänzen weiterer Eingabegeräte mit entsprechenden	Benutzer können keine weiteren Eingabegeräte ergänzen → Response nicht realisiert	2	Neue Schnittstellen und neue Controller	3	[+] Modifizierbarkeit [?] Performanz	ja
			5	Nicht-Existenz einer Regelung zum Ergänzen weiterer Ausgabegeräte mit entsprechenden	Benutzer können das System nicht mit weiteren Anwendungen oder anderen Geräte verwenden → Response nicht realisiert	2	Änderung des Interfaces und/oder des Protokolls, verschiedene Architekturen möglich (Frage nach gleichzeitiger Bedienung mehrerer Clients)	3	[+] Modifizierbarkeit [?] Performanz	ja

Abbildung 36: Entscheidungen S4 und S5 betreffen das Thema Erweiterbarkeit

Erläuterung: Es ist noch nicht konzipiert, wie neue Ausgabegeräte angeschlossen werden können (S5). Es gibt noch keine Regelung in der Softwarearchitektur, wie Eingabe-Clients angeschlossen werden können (S4).

Hinweise auf mögliche Probleme bei der Benutzung:

(-) Fehlende Schnittstellen und die Notwendigkeit von spezifischen `SMSController`-Komponenten für externe Eingabegeräte führen dazu, dass derzeit keine Eingabe-Clients angeschlossen werden können: Aktuell sammelt `SMSLib` Rohdaten der Beschleunigungsaufnehmer, leitet die Daten an `SMSController` weiter, der mit `PatternEngine` kooperiert, um Rohdaten analysieren zu lassen und Events zu erhalten, die wiederum an `SMSServer` versendet werden. Um das System um weitere Eingabemöglichkeiten erweitern zu können, müssten die Komponenten `SMSLib`, `PatternEngine` und `SMSController` gekapselt und auf ein externes Geräten verlegt werden können, entsprechende Schnittstellen zwischen `SMSController` und `SMSServer` müssten vorhanden sein. `SMSController` müsste für bestimmte Endgeräte separat erstellt werden, wobei es dann voraussichtlich auch notwendig wäre, dass jeweils GUI-Komponenten für die Nutzer/Administratoren eines SMS-Controllers erstellt werden.

(-) Schnittstellen und die Verteilung des `SMSClient` sowie das schon beschriebene Kommunikationsproblem verhindern, dass derzeit weitere Ausgabeprogramme

oder Ausgabegeräte angeschlossen werden können: Um neue Programme zu ergänzen, müssten neue Anwendungs-Clients erstellt und das entsprechende Interface angepasst werden. Dies würde prinzipiell noch im Rahmen der aktuellen Architektur möglich sein (siehe Thema Kommunikation S. 104). Doch weitere Ausgabegeräte können noch nicht angeschlossen werden: dafür müsste, nach Ansicht des Architekten, die Komponente `SMSClient` auf dem Mac positioniert werden, um von dort aus verschiedene Ausgabegeräte anzusprechen.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(-) Performanz: Sobald mehrere Interaktionsgeräte oder Ausgabegeräte, auf denen Anwendungen laufen, ergänzt werden, ist das Kommunikationsprotokoll TCP/IP ungeeignet (siehe Diskussion zum Thema Kommunikation, S. 104).

(-) Erweiterbarkeit: Die Erweiterbarkeit um externe Eingabe- und/oder Eingabegeräte ist noch nicht gewährleistet.

Mögliche Modifikationen:

- Kapselung von Komponenten in eine Eingabe-Komponente und Netzwerkmonitoring soll es ermöglichen, externe Eingabegeräte zu ergänzen: Um externe Eingabegerät zu ergänzen, müssten auf dem externen Eingabegerät eine oder mehrere neue Komponenten installiert werden (unterschiedliche Möglichkeiten). Diese müsste(n) (1) die Rohdaten von Bewegungssensoren mittels einer entsprechenden Bibliothek liefern können und außerdem (2) Netzwerkverbindungen etablieren, überwachen und verwalten können. Diese Änderung betrifft verschiedene Komponenten und ist sehr aufwendig, da es zudem verschiedene Möglichkeiten gibt, die Komponenten zu verteilen. Es ist noch zu bewerten, welche Möglichkeit präferiert wird, weshalb der Änderungsaufwand unbekannt ist (Änderungskomplexität 3).
- Eine Umverteilung der Komponente `SMSClient` auf den Mac und das Protokoll UDP können es ermöglichen, weitere Ausgabegeräte anzuschließen: Um Ausgabegeräte zu ergänzen, sollte die Aufbereitung der Daten auf dem Mac geschehen. Die Komponente `SMSClient`, die aktuell auf einem PC läuft, wird also auf den Mac „verlegt“ und ist deshalb neu zu implementieren. Dieser Teil der Anwendung auf dem PC ist anzupassen. Das bedeutet, `GEOutputClient` und `PPTOutputClient`, die auf verschiedenen Ausgabegeräten laufen, sollen sich an einem Mac anmelden und von dort aufbereitete Daten erhalten. Aufgrund des zu erwartenden sehr hohen Kommunikationsaufwandes und Verzögerungen bei TCP/IP, könnte, wie schon erwähnt (S. 104), UDP genutzt werden. Diese Änderung betrifft verschiedene Komponenten, ist also aufwendig und sehr komplex. Die verschiedenen Optionen müssen noch bewertet werden, so dass auch hier Änderungsaufwände noch unbekannt sind (Änderungskomplexität 3).

5.2.6.4 *Sonstiges*

Abbildung 37 fasst Kommentare zusammen, die während der Analyse nicht weiter diskutiert wurden.

Beim Interaktionsszenario Add New Input/Output Device betrifft dies weitere Anforderungen, die in die nächste Analyse eingehen sollten. Kommentare beim Interaktionsszenario Recovering from Failure betreffen zudem eine Frage, die im nächsten Analysedurchgang beachtet werden soll sowie eine neue Anforderung.

Szenario	Kommentar	Rubrik
Cancel	keine	ohne
Add New Input/Output Device	- Vorgesehen, dass die Eingabe modularisiert wird und dass jede Art von Bewegungsdaten aufgenommen und interpretiert werden kann; - außerdem Ziel, dass Benutzer zwischen den Programmen wechseln können	Offene Fragen Softwarearchitektur
Feedback	keine	ohne
Recovering from Failure	- Kann der Status programmintern zwischengespeichert werden? Wie definiert und aktualisiert man den Status, wenn nicht? - beim Environment im Museum muss der SMS-Controller selbständig starten	Offene Fragen Softwarearchitektur

Abbildung 37: Während der Softwarearchitekturanalyse wurden weitere Kommentare notiert.

Szenario	Fragen an Usability-Evaluation	Priorisierung
Cancel	Beobachten und bewerten, wie Benutzer die Verzögerung bei Google Earth wahrnehmen.	ASL 1
Add New Input/Output Device	Test, ob und wie einfach Administratoren I/O-Geräte anschließen können.	ASL 2
Feedback	Response Measure testen „Users are supplied with all necessary data. (U) Users perceive all necessary data. (U) The data perceived by the users are up to date. (U)“	ASL 1
Recovering from Failure	Test eines Evaluationsszenarios, in dem Netzwerkprobleme auftreten.	ASL 2

Abbildung 38: Bewertung der Interaktionsszenarios und der darin gestellten Fragen an eine Usability-Evaluation

5.2.7 *Durchführung der Fallstudie SYM: SATURN-Phase 5*

Bewerten der Methode und ihrer Werkzeuge

Zuerst wurden alle SATURN-Dokumente und Ergebnisse archiviert, d.h. der Analysekontext SYM, Interaktionsszenario-Auswahl SYM, Evaluationsformular SYM, Ergebnisse der SATURN von SYM. Danach wurde ein neues Interaktionsszenario Add New Input/Output Device erstellt.

Die Befragung erfolgte mündlich und das positive Feedback umfasste folgende Aspekte:

- Die Analyse war effizient und führte zu interessanten Erkenntnissen und neuen Anforderungen.
- Der Architekt hat nun das aktuelle Design im Überblick.
- Die Diskussion der Alternativen war sehr hilfreich und gab wertvolle Impulse für die nächste Version, die sofort erarbeitet wird.
- Die Auswahl der Interaktionsszenarios war sehr gut. Es wurden viele aktuelle offene Fragen aufgedeckt, begründet, ausgesprochen, dokumentiert.
- Insbesondere die Anforderung, weitere Eingabe- und Ausgabegeräte zu unterstützen, ist besonders sinnvoll für dieses System. Sie wurde bei der Diskussion und der Auswahl der Interaktionsszenarios bestimmt. Bei der Untersuchung des Interaktionsszenarios wurden die derzeitigen Hindernisse verdeutlicht.

Verbesserungsvorschläge waren:

- Vorgeschlagen wurde eine Klassifikation der Interaktionsszenarios nach Interaktionskategorien.
- Es wird eine gemeinsame Diskussion jedes von einer Person ausgewählten Interaktionsszenarios vorgeschlagen, nachdem jede Person eine eigene Auswahl getroffen hat. Die Interaktionsszenarios sollen nach einer Diskussion ihrer Bedeutung ausgewählt werden, nicht nach einer Abstimmung, wie beispielsweise

bei ATAM. Werden zu viele Interaktionsszenarios ausgewählt (Zeitaufwand, gewünschter inhaltlicher Fokus, Themen bei der Entwicklung zum Zeitpunkt der Analyse), soll die Methode in mehreren Durchgängen durchlaufen werden können.

- Es sind nicht immer Modelle oder Patterns nötig, so dass deren Einsatz nicht erzwungen werden sollte. Sie geben allerdings gute Hinweise auf Problemfelder und mögliche Umsetzungen sowie alternative Lösungen. Referenziert werden sollten sie also weiterhin, ohne Zwang. Die eigentliche Analyse erfolgt ohnehin durch eine Diskussion der Use Cases und ihrer Unterstützung für die Anforderung, die durch das betreffende Interaktionsszenario ausgedrückt wird.
- Die offenen Fragen könnten, sofern sie inhaltlich dazu gehören, bei den Themen angesprochen werden. Es besteht die Gefahr, dass sie übersehen werden.
- Die Qualität der Interaktionsszenarios muss gewährleistet sein, ein Prozess zur Erstellung gültiger Interaktionsszenarios muss eingehalten werden. Dazu müsste dieser zuerst thematisiert werden.
- Es gibt offenbar Interaktionsszenarios, die nicht nur Usability betreffen, sondern auch andere Qualitätseigenschaften. Diese sollten genannt werden und mit Hilfe jeweils passender Methoden untersucht werden.

5.2.8 *Nutzertest*

5.2.8.1 *Durchführung des Nutzertests*

Da während des Nutzertests (synonym verwendet für User-Evaluation) Faktoren getestet werden, die während der Nutzung von SYM in einem Seminarraum auftreten, wurde die User-Evaluation für SYM als in vitro Evaluation aufgesetzt; also nicht als Feld-, sondern als Laborversuch.

Die Nutzungsfaktoren (Umwelt, Benutzer, Aufgabe, Endgerät und Anwendung) sind während der Konzeption der User-Evaluation besonders wichtig, denn sie sind die Basis für die Identifikation der zu analysierenden Parameter. Beispielsweise wurden die Szenarios für den Test von ihnen abgeleitet; und zu evaluierende Interaktionen wurden an die Benutzung von Google Earth im Museum angelehnt.

Für die Evaluation wurden 12 Testbenutzer ausgewählt, um damit ein Resultat zu erhalten, dessen statistische Relevanz für eine formative Usability-Evaluation ausreichend ist [BBC+03]. Von den Testpersonen waren 58% weiblich und 42% männlich, das durchschnittliche Alter betrug 31 Jahre. Alle Sitzungen wurden auf Video aufgenommen und später analysiert. Dazu wurde ein spezielles Evaluations-Framework verwendet, das speziell für mobile Anwendungen entwickelt wurde [Gri09].

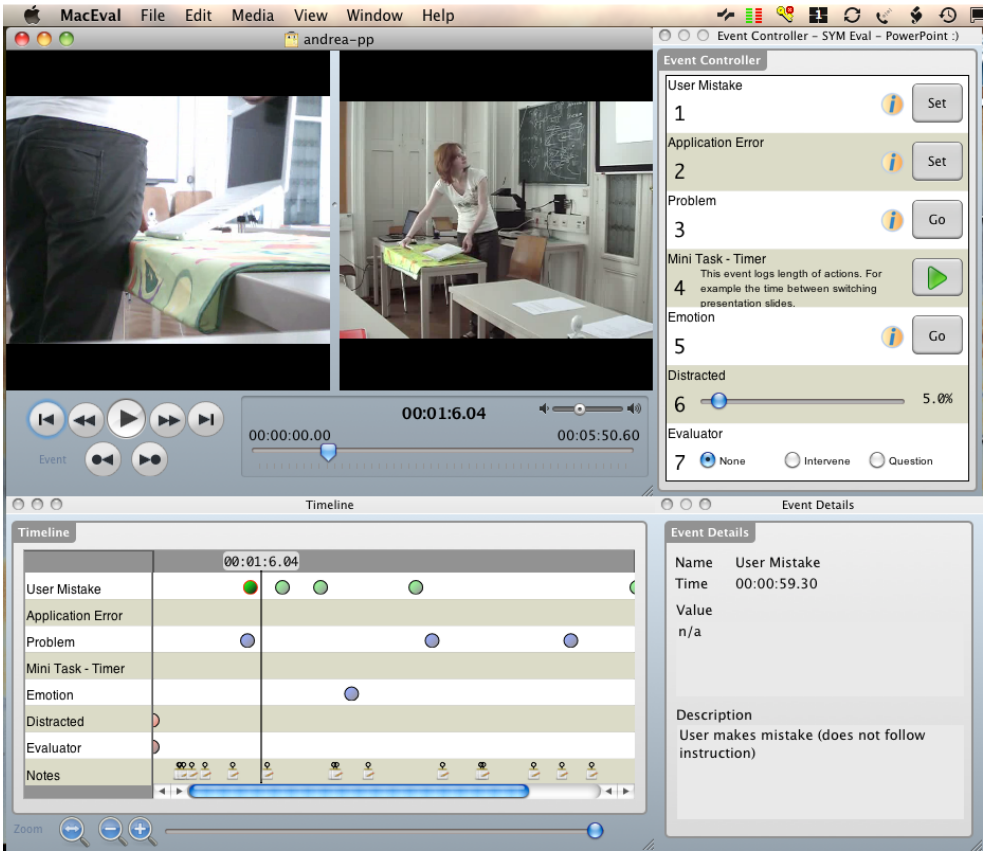

Abbildung 39: Analyse der Evaluation mit MacEval

Das Tool MacEval[2] (siehe Abbildung 5.2.8.1) ist Teil des verwendeten Evaluations-Frameworks, mit dem Sitzungen aufgenommen, abgespielt und einzelne Geschehnisse annotiert werden können. Im Rahmen einer Inhaltsanalyse wurden die Sitzungen wiederholt angesehen und die Aktionen der Benutzer anhand der Parameter in Tabelle

2 MacEval – `http://www.tomgrill.info/maceval`

5.2.8.1 evaluiert. Im Anschluss daran wurden die gesammelten Daten exportiert und in Microsoft Excel statistisch ausgewertet.

Parameter	*Beschreibung*
Navigations-problem	Hatten die Benutzer Probleme bei der Navigation?
Verständnis	Verstanden die Benutzer, wie das SYM-Interaktionsgerät in Kombination mit der zu kontrollierenden Anwendung verwendet werden muss?
Reaktion der Anwendung	Reagiert die Anwendung gut?
Interaktions-art	Welche Interaktionsarten wurden verwendet und wie ist (statistisch gesehen) deren Verteilung?
Modi	Verstanden die Benutzer die Nutzung von Modi? Waren sie in der Lage, richtig zwischen den Modi zu wechseln?
Aufgabe	Wie erfolgreich waren die Benutzer beim Erledigen ihrer Aufgaben?
Lernkurve	Wie steil war die observierte Lernkurve?
Zufriedenheit	Wie zufrieden waren die Benutzer während der Interaktion mit SYM?
Intuition	Wie intuitiv wurde die Interaktion mit SYM von den Nutzern verwendet?

Tabelle 30: Beobachtete Parameter

Während der Evaluation führten die Benutzer zwei Aufgabenblöcke mit einer durchschnittlichen Dauer von jeweils fünf Minuten aus. Als erstes mussten sie, mit SYM als Interaktionsgerät und einem vorbereiteten Powerpoint-Foliensatz, einen Vortrag halten. Drei Teilaufgaben waren so gestellt, dass alle Funktionen der Navigation verwendet werden mussten. Der zweite Aufgabenblock enthielt drei Aufgaben, bei denen die verschiedenen Möglichkeiten der Google Earth Navigation eingesetzt werden mussten. Nach der Durchführung der Evaluation wurden die Resultate hinsichtlich der Parameter ausgewertet.

5.2.8.2 *Resultate des Nutzertests*

Die Abbildung 40 zeigt anhand eines Boxplots (Kastengrafik) die Verteilung der statistischen Daten. Boxplots zeigen die Streuungs- und Lagemaße Median, zwei Quartile, die zwei Extremwerte und Ausreißer an. Boxen repräsentieren jeweils, in welchem Bereich 50% der Daten liegen. Der Median teilt die Box in zwei Hälften.

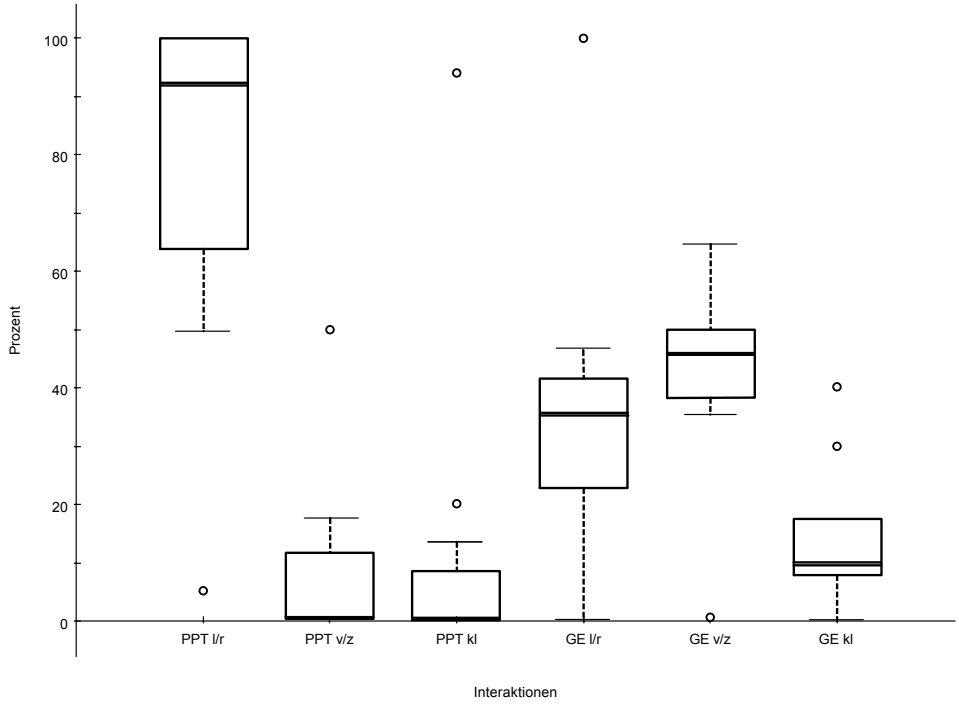

Abbildung 40: Verteilung der Interaktionstechniken bei der Interaktion mit PowerPoint und Google Earth

Bei der Evaluation mit PowerPoint wurden Klopfen und Bewegen des Gerätes nach vorn und hinten selten durchgeführt (nur ca. 20% aller Interaktionen). Die meisten Benutzer verließen sich auf das Kippen nach rechts oder links zur Navigation. Es wurde nicht erkannt, dass auch das Klopfen zur Vorwärtsnavigation verwendet werden konnte. Bei der Evaluation mit Google Earth wurden von den Nutzern verschiedene Interaktionstechniken eingesetzt: Zu 34,25% wurde die links/rechts-Navigation eingesetzt, zu 40,42% die Bewegung vor/zurück. Klopfen betraf 13% der Interaktionen, weitere Techniken wurden nicht beobachtet. Die beim Google Earth eingesetzten Techniken variieren mehr als beim PowerPoint, dies entspricht auch den Aufgaben und typischen Interaktionen der Anwendungen.

Die Tabelle 31 zeigt die univariate Varianzanalyse (ANOVA).

Unabhängige Variable	*Abhängige Variable*	*p-Wert*	*Signifikanz*
PowerPoint			
Navigationsproblem	Verstehen	0,000004878	höchst signifikant
Navigationsproblem	Antwortverhalten	0,0006382	höchst signifikant
Antwortverhalten	Verstehen	0,00007546	höchst signifikant
GoogleEarth			
Navigationsproblem	Verstehen	0,2346	nicht signifikant
Navigationsproblem	Antwortverhalten	0,005997	sehr signifikant
Antwortverhalten	Verstehen	0,224	nicht signifikant

Tabelle 31: Varianzanalyse der Nutzungskontextfaktoren

Beim Vergleich verschiedener unabhängiger Werte zeigt sich ein starker Zusammenhang zwischen allen Kombinationen der beobachteten Parameter für die Evaluation mit PowerPoint; hinsichtlich Google Earth besteht ein enger Zusammenhang zwischen Navigationsproblem und dem Antwortverhalten der Anwendung. Dieses Problem trat während der Evaluation mit beiden Anwendungen auf. Daraus kann geschlussfolgert werden, dass diese zwei Parameter miteinander korrelieren und dass es ein allgemeines Navigationsproblem gibt, das vom Antwortverhalten der Anwendung verursacht wird.

Die Erlernbarkeit der einzelnen Interaktionen war gut. Ausgehend von der Verteilung der eingesetzten Interaktionen konnten die Nutzer schneller lernen, mit PowerPoint zu interagieren als mit Google Earth.

Der Zusammenhang zwischen Navigationsproblemen und dem Antwortverhalten der Anwendung kann durch die kurze Aufmerksamkeitsspanne bei mobiler Nutzung (siehe Faktoren des mobilen Nutzungskontexts, Abschnitt 4.1.2, Seite 51) erklärt werden; allerdings deuten die Kommentare der Benutzer darauf hin, dass sie kaum Kontrolle über Google Earth hatten.

Durch die Analyse mit SATURN wurde die Ursache für dieses Verständnisproblem aufgedeckt: der `GEOutputClient` zeigt nicht an, wann Google Earth eine Interaktion abarbeitet (siehe Abschnitt 5.2.6.3, S. 106). Die Benutzer können nicht wissen, dass sich ihre Events in einer Warteschlange befinden und nach und nach abgearbeitet werden.

5.2.8.3 *Resultate von SATURN und Nutzertest*

Die Resultate von SATURN und der User-Evaluation wurden anhand der in Tabelle 32 erläuterten vier verschiedenen Möglichkeiten klassifiziert.

Typ	*Problem ermittelt via*		*Argumentation*
	SAA	*UE*	
1	✓	*	Problem ermittelt durch SATURN, nicht durch die User-Evaluation *Beispiel:* Der SYM-Prototyp ist noch nicht dazu in der Lage, mehrere Eingabe- oder Ausgabegeräte zu bedienen.
2	*	✓	Problem ermittelt durch die User-Evaluation, nicht durch SATURN *Beispiel:* Die Antwortzeit des Systems war für die Benutzer unangemessen.
3	✓	✓	Problem sowohl durch SATURN als auch durch die User-Evaluation ermittelt *Beispiel:* SATURN ermittelte, dass die Statusanzeige den Status des Sensors anzeigt, aber nicht den Status der Anwendung. Die User-Evaluation fand heraus, dass die Benutzer durch die Statusanzeige irritiert waren.
4	*	*	Problem wurde weder durch SATURN noch durch die User-Evaluation ermittelt.

Tabelle 32: Vergleich der Resultate einer Software-Architektur-Analyse und der Usability-Evaluation

Die folgenden Probleme wurden miteinander in Verbindung gebracht. Sie wurden diskutiert und führten zu Vorschlägen und der Bestimmung von offenen Punkten.

- Die Events, die aktuell abgearbeitet werden, werden nicht angezeigt (siehe Thema Antwortverhalten in Abschnitt 5.2.6.3, S. 106). Die User-Evaluation zeigte, dass die Benutzer sehr irritiert waren (Navigationsproblem wegen Antwortverhalten). In SATURN wird unter anderem vorgeschlagen, das Problem durch die Änderung des Interfaces und der Komponenten `PPTOutputClient/GEOutputClient` zu beheben und/oder eine andere Queue einzusetzen.
- Weitere Eingabe-/Ausgabegeräte sollten ergänzt werden, damit die Benutzer das für sie passende Endgerät verwenden können (siehe Thema Erweiterbarkeit in Abschnitt 5.2.6.3, S. 107). Diese Anforderung wurde noch nicht umgesetzt und konnte somit nicht bei der User-Evaluation getestet werden. In SATURN wird vorgeschlagen, die Wahl von TCP/IP als Protokoll und die Verteilung der Komponente `SMSClient` zu überdenken.
- Es ist nicht möglich, automatisch Verbindungsabbrüche zu erkennen und darauf zu reagieren (siehe Thema Kommunikation in Abschnitt 5.2.6.3, S. 104). Das Managen der drahtlosen Netzwerkverbindung soll noch integriert werden.
- Die User-Evaluation fand eine Vielzahl an Fehlern, die während der Interaktion mit dem Gerät für die Benutzer entstanden. Es wurden beispielsweise Fehler aufgrund von Missverständnissen (Error) und versehentliche Fehler (Slips) gefunden. Diese Probleme wurden als Typ 2-Probleme klassifiziert. Anhand der Ergebnisse kann nun festgestellt werden, welche Interaktionen sofort intuitiv verwendet wurden. Diese sollten weiterentwickelt werden, während die anderen verworfen werden können.
- Die User-Evaluation ermittelte Folgeprobleme, die sich aus kleinen Fehlern ergaben. Auch dies sind Probleme von Typ 2.

- Die User-Evaluation ermittelte verschiedene Herangehensweisen an Probleme, so dass andere Denkmuster sichtbar wurden.
- Insgesamt konnte die User-Evaluation die Zufriedenheit der Benutzer beobachten. Abhängig davon konnte ermittelt werden, wie ernst die Usability-Probleme waren.

Die Probleme von Typ 1 und Typ 2 können dahingehend überprüft werden, ob sie Informationen oder Fragestellungen für die jeweils andere Methode bereithalten. Die Tabellen 41 und 42 zeigen die Details zum Mapping.

Problem UE	SATURN: szenario-beeinflussende Architektur-entscheidung	Typ	IS für kommenden SATURN-Durchgang
Notebook reagierte nicht sofort	ankommender Event direkt zur UI gesendet, statt vom UI durchgeführter Event (S6)	3	Benutzer-Tempo (31)
Notebook reagierte nicht sofort auf den „Stop“-Event	System beendete aktuelle Events nacheinander, statt ein Stop durchzuführen (S7)	3	Benutzer-Tempo (31), Abbrechen (4)
Notebook reagierte nicht sofort	Performanzproblem wegen der Entscheidung TCP/IP zu nutzen (S8)	3	Konfliktfreie Nebenläufigkeit (24), Geräteunabhängigkeit (25)
Notebook reagierte nicht auf das Klopfen		2	Multimodale Interaktion mit funktionaler Konsistenz (neues IS?)
Nutzer weiß nicht, was passiert. Aktueller Systemstatus wird nicht korrekt angezeigt.	ankommender Event direkt zur UI gesendet, statt vom UI durchgeführter Event (S6)	3	Fortschrittsanzeige (32)
Nutzer hat Eindruck, dass System nicht auf seine Eingaben reagiert		2	Wiederherstellung nach Betriebsausfall (26)
Nutzer ist unzufrieden mit Interaktionsdesign		2	
Nutzer weiß nicht, wie Modus zu ändern ist		2	Modi und Profile (19)
Nutzer findet die Richtung nicht. Er probiert vor/zurück statt rechts/links, er probiert viele verschiedene andere Möglichkeiten (Schwenken des Notebooks)		2	Mehrstufige Hilfe (47)
Nutzer verwendet vor/zurück Interaktion statt rechts/links		2	
Nutzer hebt Notebook an. Hat nicht funktioniert.		2	
Nutzer weiß nicht, wie er überhaupt navigieren soll.		2	
Nutzer wusste nicht, warum das System den Modus änderte		2	Wiederherstellung nach Betriebsausfall (26)
Evaluator musste erklären, wie interagiert werden soll		2	Aufforderung zur Fehlerbehebung (15), Fehlermeldungen (43), Wizard (44), Mehrstufige Hilfe (47)
Applikation scheint nicht 100%-ig korrekt kalibriert zu sein, Laptop wurde gekippt während es angehoben war und das System reagierte nicht auf diese Interaktion.		2	Modi und Profile (19)
Applikation scheint nicht korrekt kalibriert zu sein.		2	Modi und Profile (19)

Abbildung 41: Mapping der Ergebnisse von User-Evaluation und SATURN, Seite 1

Problem UE	SATURN: szenario-beeinflussende Architektur-entscheidung	Typ	IS für kommenden SATURN-Durchgang
Applikation wechselt Modus automatisch		2	Benutzer-Tempo (31)
Combination von Navigation +Modus ändern ist für Nutzer anspruchsvoll		2	
Lärm stört die Nutzerinteraktion		2	
Klopfen hat unterschiedliche Funktionen in unterschiedlichen Anwendungen		2	Vertrautes Aussehen und Verhalten (34)
	Nicht-Existenz einer Regelung zum Ergänzen weiterer Eingabegeräte (S4)	1	
	Nicht-Existenz einer Regelung zum Ergänzen weiterer Ausgabegeräte (S5)	1	
	Nicht-Existenz einer Vorrichtung zum Monitoring der Verbindung (S9)	1	
	Gesamt: 23		
	Typ 1 (SAA): 3 (13,0 %)		**Typ 2.1: 10 (43,5%)**
	Typ 2 (UE): 16 (69,6)		**Typ 2.2: 6 (26,1%)**
	Typ 3 (SAA&UE): 4 (17,4%)		

Abbildung 42: Mapping der Ergebnisse von User-Evaluation und SATURN, Seite 2

5.2.8.4 *Rückblick*

Die Verteilung der identifizierten Probleme ist wie folgt: 30% der identifizierten Usability-Probleme wurden durch SATURN und 87% durch die User-Evaluation ermittelt. Allerdings wurde nur eine minimale Anzahl von vier Interaktionsszenarios analysiert. Dennoch weist dieses Ergebnis darauf hin, dass eine User-Evaluation im Vorfeld von SATURN oder abwechselnd mit SATURN durchgeführt werden sollte.

Es stellt sich die Frage, inwiefern die Resultate der User-Evaluation und der Softwarearchitekturanalyse miteinander in Beziehung stehen. Auf den ersten Blick erscheint es, als würden sich die Probleme zu 17% überlappen. Mit der Methode zur Kombination von SAA und der Usability-Evaluation analysierten, verglichen und diskutierten wir die Ergebnisse. Alle Anforderungen von Typ 1 und Typ 3 repräsentieren Probleme, deren Ursachen offensichtlich potentiell in der Softwarearchitektur liegen. Die Ergebnisse von Typ 2 können in zwei Kategorien unterteilt werden.

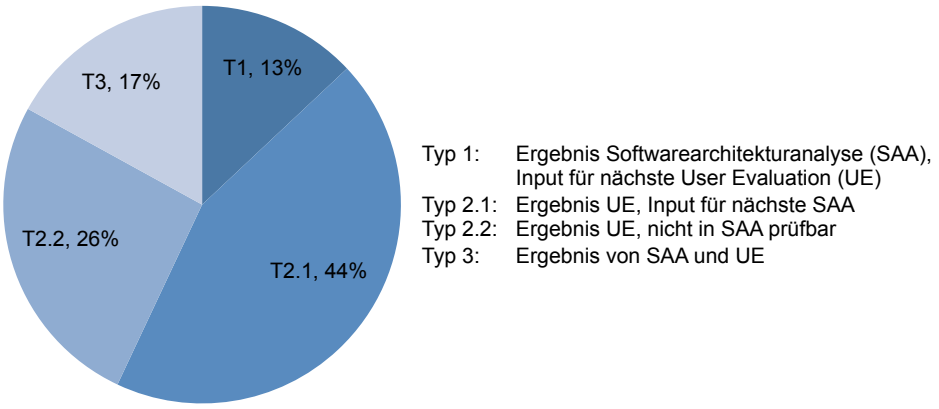

Abbildung 43: Ergebnisse für zukünftige Entwicklungszyklen

Die erste Kategorie (Typ 2.1) zeigt Ergebnisse, die einen potentiellen Einfluss auf die Softwarearchitektur haben und deshalb in der nächsten Iteration der SA-Analyse beachtet werden müssen. Probleme vom Typ 2.1. werden anhand eines Mappings von Problemen auf die Interaktionsszenarios in der Wissensbasis ermittelt. Die zweite Kategorie (Typ 2.2) sind Usability-Probleme, die nicht durch die Softwarearchitektur beeinflusst werden können. Es ergibt sich, dass 44% der Ergebnisse von Typ 2.1 sind, da 15 existierende Interaktionsszenarios aus CASSINI auf die Probleme abgebildet werden konnten. Die übrigen 26% der Ergebnisse konnten nicht zugeordnet werden, sie sind inhaltlich von Typ 2.2, d.h. sie betreffen nur Fehler der Benutzer. Somit zeigen die Ergebnisse, dass 74% der Usability-Probleme Ursachen in der Softwarearchitektur zugeordnet werden konnten.

Die Ergebnisse von Typ 1 (13%) sollten für die kommenden User-Evaluationen verwendet werden, damit die in der Architektur ermittelten problematischen Interaktionsszenarios in allen folgenden Benutzertests überprüft werden.

Diese Kombination hat wesentliche Vorteile:

- Es werden mehr Probleme gefunden als mit jeweils nur einer Methode.
- Durch das Mapping der Ergebnisse werden sofort Ursache-Wirkungs-Paare ermittelt, so dass zeitaufwendiges Suchen entfällt.
- Es ist möglich zu bestimmen, wie kritisch es für die Usability ist, dass ein Interaktionszenario nicht oder nur teilweise durchführbar ist. Die Annahmen, die während der Architekturanalyse gemacht wurden, können also überprüft werden. Es wird damit das Risiko vermindert, dass komplexe Probleme in der Architektur, die keine große Auswirkung auf die Usability haben, eine zu hohe Priorität erhalten und dass genau die Probleme zuerst erhoben werden, die tatsächlich schwerwiegend sind. Die Priorisierung der Interaktionsszenarios kann gegebenenfalls geändert werden.
- Anhand der Diskussion der Probleme von Typ 2 können Interaktionsszenarios ausgewählt werden, deren Response nicht umgesetzt wurde. Diese können sofort als Interaktionsszenarios festgelegt und auf architektonische Ursachen hin untersucht werden. Es erübrigt sich dann die Auswahl und deren Diskussion. Damit ergibt sich ab dem zweiten Durchgang der Kombination SATURN und User-Evaluation eine Effizienzsteigerung.

5.2.9 *Beurteilung*

5.2.9.1 *Machbarkeit*

Die Analyse zeigte, dass der wissenschaftliche Prototyp Interaktionsmuster erkennt und weiterleitet. Dennoch konnte keines der vier Interaktionsszenarios (Add New Input/Output Device, Recovering from Failure, Abbrechen, Feedback) komplett positiv bewertet werden. Wechselwirkungen mit den Qualitätsmerkmalen Modifizierbarkeit/Erweiterbarkeit, Responsivität/Performanz, Effizienz und Fehlertoleranz wurden offengelegt.

Mit der Methode SATURN wurde also ermittelt, dass die untersuchten Anforderungen, repräsentiert durch die Interaktionsszenarios, architektonisch nur teilweise unterstützt werden und weshalb dies so ist. Die Machbarkeit von SATURN ist für diese Fallstudie also bestätigt.

Zusätzlich wurde SATURN mit einer User-Evaluation kombiniert. Fragen der einen Methode beantworteten die Ergebnisse der jeweils anderen Methode: so wurden architektonische Ursachen von Usability-Problemen ermittelt, Ergebnisse von SATURN bestätigt und ihre Priorität durch die Schwere des beobachteten Usability-Problems bestimmt (Probleme wegen des verzögerten Feedbacks). Ergebnisse, die nicht sofort aufeinander abgebildet werden konnten, wurden diskutiert und können nun teilweise als neue Fragestellungen in die Evaluation und Analyse zurückfließen. Damit können noch genauere Informationen ermittelt werden, so dass die Qualität und damit die Verwertbarkeit der Resultate optimiert wird. Die Ergebnisse haben damit eine höhere Quantität sowie Qualität und können so im Entwicklungsprozess integriert werden, dass schneller eine gute Softwarequalität und Usability erreicht werden können.

5.2.9.2 *Nützlichkeit*

Es stellt sich die Frage, ob der Architekt von SYM es nützlich fand, diese Methode durchzuführen. Im Rahmen der Diagnose[3] wurden folgende Ziele festgelegt: Vorbereiten der neuen Version des Prototypen; Verstehen, wie Usability-Anforderungen in der aktuellen Version architektonisch unterstützt werden (Abschnitt 5.2.1, S. 86).

Das Feedback des Architekten in der Phase 4 „Retrospektive“ zeigt, dass er mit den Ergebnissen zufrieden war, denn beide Ziele wurden erreicht: er hat nun das aktuelle Design im Überblick und die Diskussionen gaben ihm wertvolle Ideen für die nächste Version, die er sofort im Anschluss erarbeiten möchte. Gerade die Wahl der Interaktionsszenarios, die Diskussion von Alternativen und die Überlegungen dazu, wie weitere Ein- und Ausgabegeräte unterstützt werden können, fand er sehr hilfreich (letzteres war eine neue Anforderung). Aus seiner Sicht war es also nützlich, diese Architekturanalyse durchzuführen. (Abschnitt 5.2.7, S. 109)

Nützlich waren die Ergebnisse außerdem für die Konzeption eines Nutzertests. Die Anforderungserhebung wurde weiterverwendet, um eine User-Evaluation zu konzipieren und dann Probleme bei der Benutzung aufzudecken. Die wissenschaftliche Fragestellung war, wie eine Softwarearchitekturanalyse mit einem Nutzertest kombiniert werden kann.

Im Zuge des Vergleichs der Ergebnisse von SATURN und dem Nutzertest zeigte sich, dass ein zentrales Problem der Benutzbarkeit der Anwendung SYM (das Antwortverhalten der Anwendung) durch die Architekturanalyse ermittelt wurde. Weitere Themen waren die Kommunikation und Erweiterbarkeit. Der Nutzertest wiederum verwies auf weitere - aus Anwendersicht nützliche und zu prüfende - Interaktionsszenarios hin (z.B. Wiederherstellung nach Betriebsausfalll, Modi und Profile, Benutzer-Tempo, siehe Abbildungen 41 auf S. 117 und 42 auf S. 118). Mit diesem Vorgehen wurde

3 siehe Canonical-Action-Research-Zyklus in Abschnitt 5.1.3 ab S. 86

eine ganzheitliche Betrachtung von Usability erreicht, mit der qualitativ bessere und quantitativ mehr Probleme ermittelt wurden als nur mit einer Methode.

5.2.9.3 *Verständnis*

VORGEHENSMODELL Die einzelnen Arbeitsschritte wurden sofort nach einer kurzen Erklärung verstanden, da sich nach Aussage des Architekten die Methode logisch aufbaut und sich an der typischen Vorgehensweise der szenario-basierten Architekturanalysen orientiert. Zudem war dem Architekten die Einbeziehung des Geschäfts- und Nutzungskontexts aus dem Fachgebiet MMI bekannt.

Es wurde ermittelt, dass die Zusammenhänge zwischen Nutzungskontext und Interaktionsszenarios genauer herausgestellt werden sollten und dass die Beschreibung von Interaktionsszenarios angeleitet werden sollte.

INTERAKTIONSSZENARIO Außerdem wurde der Begriff „Interaktionsszenario“ diskutiert, da Szenarios in vielen Gebieten eingesetzt werden; insbesondere bei Nutzertests, um den Probanden eine umfassende Aufgabe vorzugeben, die auch als Szenario bezeichnet werden kann. Solch ein Szenario bei einer User-Evaluation umfasst dann aber nicht nur eine, sondern mehrere einzelne Interaktionen einer Testperson. Ebenso wird in solchen Szenarios nicht die Reaktion eines Systems thematisiert. Die in SATURN verwendeten Interaktionsszenarios beschreiben die Response des Systems auf eine einzelne Interaktion. Nachdem dieser Unterschied erläutert und begründet wurde, waren die anfänglichen Verständnisprobleme beseitigt.

UNTERSTÜTZUNG DURCH HILFSMITTEL Während der Durchführung der Methode wurden Hilfsmittel erstellt, d. h. grob strukturierte Vorlagen wurden während der Fallstudie den praktischen Anforderungen angepasst. Diese Unterlagen können verfeinert werden und als Vorlagen von einzelnen Tabellen, Checklisten dienen. Somit wird auch gewährleistet, dass mit Hilfe dieser Gedächtnisstützen die Methode entsprechend ihrer Vorgabe bearbeitet wird.

KOMBINATION DES NUTZERTESTS UND SATURN Durch die Kombination der Methode SATURN mit dem Nutzertest wurde eine größere Anzahl von Problemen ermittelt als nur mit einer einzelnen Methode.

5.2.10 *Reflexion*

5.2.10.1 *Aufwand*

Die Methode SATURN (ohne den Nutzertest) wurde in vier Personentagen durchgeführt: zwei Personen arbeiteten einen Tag in Linz und je zwei halbe Tage über Internettelefonie mit Bildschirmübertragung miteinander. Dieser Zeitaufwand war geringer als erwartet, rückblickend jedoch aufgrund der Architekturbeschreibung zu Beginn der Methodendurchführung größer als nötig.

Die Softwarearchitektur war nicht gut dokumentiert. Der Analyst musste häufig nachfragen, um das Design zu verstehen. Eine minimale Dokumentation wurde deshalb am Anfang noch in SATURN-Phase 1 erarbeitet: CRC-Karten, das Verteilungsdiagramm, das Kommunikationsprotokoll und die Durchführung der Nutzer-System-Interaktion waren relevant. Die Diskussionen gingen ins Detail und über rein strukturelle Fragen hinaus. Wenn es um das Verhalten des Systems geht, spielen Protokolle und Schnittstellen eine sehr große Rolle. Die SA soll daher bereits einen Status haben, der absichert, dass genug Informationen über Nutzer-System-Interaktionen vorliegen.

Damit die Methode selbst keinen unüberschaubaren Aufwand generiert, wird festgelegt, dass die Dokumentation im Vorfeld durch den Softwarearchitekten zu erstellen ist.

Sie kann, wie auch in dieser Fallstudie, gemeinsam erarbeitet werden, aber außerhalb von SATURN.

In dieser Fallstudie war der Architekt die Hauptinformationsquelle, daneben wurden Modelle zu Rate gezogen, um das Verständnis über die Architektur zu verbessern. Der Quelltext lag zwar als Code-Rahmen vor, dieser wurde aber nicht untersucht. Betrachtet wurden Aufgaben und Kollaborationen jeder Komponente und jeder Schnittstelle, die grafische Darstellung der Verteilung auf Hardware (UML Verteilungsdiagramm, UML Komponentendiagramm), wie die Kommunikation erfolgt, wie sie zwischen verteilten Komponenten abgesichert wird, welche Statusinformationen auf welche Art und Weise an das User Interface übermittelt werden.

5.2.10.2 *Aufbau und Inhalte der Methode*

Das Vorgehensmodell soll im Grundaufbau bestehen bleiben; die einzelnen Arbeitsschritte sollen klarer anleiten, auch um die Interpretation der Ergebnisse zu vereinfachen.

Während der Analyse war nicht immer klar, welche anderen Qualitätsmerkmale relevant sein können oder relevant sind. Eine Liste von Austauschbeziehungen kann die Diskussion der Austauschbeziehungen beschleunigen. Außerdem muss das Thema mit in die Beschreibung der Softwarearchitektur aufgenommen werden, damit von Anfang an festgehalten werden kann, welche Qualitätsziele zentral und damit besonders zu beachten sind.

Der Einsatz von Patterns war eine Option während der Durchführung. Sie wurden als Referenz verwendet, allerdings nicht immer. Insbesondere die Nachbereitung in Form eines Pattern-Workshops schien eine überflüssige akademische Übung. Besonders wichtig ist: Weder einfache noch umfassende Vorkenntnisse der Pattern-Literatur können bei den Teilnehmern der Analyse vorausgesetzt werden. Für die Durchführung sind Patterns auch nicht zwingend notwendig, da die Interaktionsszenarios und ihre Nutzfälle analysiert werden (nicht Pattern-basierte Fragen gestellt werden, wie es bei ATAM der Fall ist). Dennoch werden Patterns als optional einsetzbare Nachschlagewerke zur Verfügung gestellt, was aufgrund ihrer Inhalte sinnvoll ist. Der neue Prozess zur Gültigkeit der Interaktionsszenarios muss also absichern, dass zu jedem mindestens ein Pattern existiert, welches als Quelle und damit als Referenz verfügbar ist.

Der Einfluss des Forschers beschränkte sich nicht nur auf die Dokumentation, sondern es wurden auch inhaltliche Fragen gestellt, um die Softwarearchitektur besser zu verstehen. Bei der nächsten Fallstudie soll sich der Einfluss auf die Moderation beschränken.

5.2.10.3 *Wechselwirkungen*

Zusammenfassend wurden folgende Austauschbeziehungen mit anderen Qualitätsmerkmalen ermittelt.

- Modifizierbarkeit/Erweiterbarkeit: Durch die prinzipielle Möglichkeit, weitere Ausgabe-Clients zu integrieren, werden sowohl die Usability als auch die Modifizierbarkeit gefördert (S2, S3). Die Erweiterbarkeit um externe Eingabe- und/oder Eingabegeräte ist allerdings noch nicht gewährleistet (S4, S5).
- Responsivität/Performanz: Mit der Verwendung des TCP/IP-Protokolls (S8) kann es aufgrund der in Intervallen abgefragten Events zu spürbaren Verzögerungen kommen.
- Effizienz: Mehrfacheingaben werden provoziert und das Abbrechen (S7) von Interaktionen ist nicht möglich, da eine FIFO-Queue ohne priorisierte Events (S3)

eingesetzt wird und dies im User Interface nicht ersichtlich wird. Die so provozierten Mehrfacheingaben machen das System ineffizient und langsam.

- Fehlertoleranz: Netzverbindungsabbrüche führen derzeit zum Systemausfall (S9). Hier kann eine verbesserte Fehlertoleranz auch die Usability verbessern.

Die aktuellen Architekturentscheidungen behindern nicht nur, wie vorab dargestellt, die Usability, sondern auch die Qualitätsmerkmale Modifizierbarkeit/Erweiterbarkeit, Responsivität/Performanz, Effizienz und Fehlertoleranz.

5.3 FALLSTUDIE „TRAFFIC SCANNER“

5.3.1 *Diagnose*

Mit der Methode SATURN wurde eine mobile Anwendung gemeinsam mit einem Industriepartner analysiert, der an einer effizienten Durchführung und an verwertbaren Ergebnissen interessiert war. Die überarbeitete Methode, Formulare sowie 42 schriftliche bzw. über eine Website abrufbare Interaktionsszenarios lagen vor.

Die Anwendung TrafficScanner (TS) ist eine mobile Anwendung, die in Automobilen von Fahrern verwendet wird, um eine Zentrale über einen beginnenden Stau zu informieren.

Hintergrund der Analyse war das Reengineering der Softwarearchitektur dieser mobilen Anwendung, um Probleme zu ermitteln, die mit der neu zu erstellenden Version behoben werden können.

5.3.2 *Planung*

Ein Architekt und ein Analyst führten SATURN durch, ein Forscher dokumentierte. Der Architekt kannte die Softwarearchitektur nicht, hatte aber Kontakt zu dem Architekten der Anwendung; er selbst ist ein Experte auf dem Gebiet der mobilen Softwareeentwicklung. Der Analyst verfügt über Erfahrungen in den Bereichen Interaktionsdesign und Programmierung.

Das Ziel, die Ergebnisse zu veröffentlichen, wurde von allen Beteiligten von Anfang an gewünscht und unterstützt. Es wurde vereinbart, dass die Namen von beteiligten Personen und Unternehmen verborgen bleiben. Gemeinsam wurde zudem festgelegt, dass die Methode und ihre Hilfsmittel jederzeit kommentiert und (gemeinsam) verbessert werden können und sollen.

Da bei dieser Fallstudie häufig gestellte Fragen von Benutzern der Anwendung miteinbezogen wurden, wurde keine weitere User-Evaluation angestrebt; nur die Methode SATURN wurde durchgeführt.

5.3.3 *Durchführung der Fallstudie TS: SATURN-Phase 1*

5.3.3.1 *Name und Hauptverwendungszweck*

Mit der Anwendung TrafficScanner (TS) werden von Fahrern, während sie ein Fahrzeug führen, Staumeldungen an eine Zentrale gesendet. Dort werden die Daten gesammelt, mit den Meldungen anderer Fahrer abgeglichen und veröffentlicht.

Der *Hauptverwendungszweck* besteht darin, dass Autofahrer den Beginn oder das Ende eines Staus melden können. Da sich die Fahrer aber auf das Steuern des Fahrzeuges konzentrieren müssen, fragt das System bei der Unter- bzw. Überschreitung festgelegter Geschwindigkeiten nach, ob sich der Fahrer gerade an einem Staubeginn oder Stauende befindet. Die aktuelle Geschwindigkeit wird anhand der Daten eines

satellitenbasierten Navigationssystems berechnet. So müssen die Fahrer nur die Meldung des Systems bestätigen, um eine Meldung an die Zentrale zu versenden. Das *Alleinstellungsmerkmal* ist also das Staumelden „mit nur einem Klick“.

5.3.3.2 *Nutzungskontext*

Die Anwendung wird während des Autofahrens auf der Autobahn eingesetzt. Es handelt sich dabei um Autobahnen in Deutschland. Die Benutzung im Automobil geschieht während der Fahrt, deshalb ist die verfügbare Aufmerksamkeit des Fahrers für die Interaktion mit der Software extrem gering. Die Benutzer können nur mit einer Hand/nur mit einem Finger und nur kurz agieren; und das nicht nur aus praktischer, sondern auch aus rechtlicher Sicht. Die sozialen Konditionen der Umgebung sind durch Personen bestimmt, durch die ein Fahrer während der Benutzung beeinflusst werden kann, sowie auch Beifahrer, Passagiere und andere Verkehrsteilnehmer. Hinsichtlich der Konnektivität werden von der Anwendung GPS und CSD gefordert. Theoretisch mögliche Kollaborationen umfassen auch Automobile anderer Verkehrsteilnehmer, die sich vor oder nach dem eigenen Fahrzeug befinden.

Zwei Benutzertypen unterscheiden sich in erster Linie nur durch das Alter und damit zusammenhängende Einschränkungen bei der Benutzung. Die Tabelle 33 zeigt die Definition der Benutzertypen „Berufskraftfahrer“ und „Rentner“ als Zielgruppen der Anwendung.

Kategorien	**Benutzertyp 1: Berufskraftfahrer**	**Benutzertyp 2: Rentner**
Allgemeines		
Alter	Arbeitsalter 18-55	55-80
Geschlecht	Berufskraftfahrer: deutlich mehr m als w	m/w
Erfahrung im Umgang mit mobilen Endgeräten/mobilen Applikationen	Vorwissen vorausgesetzt: wie man etwas installiert, startet, bedient	Affinität zu Technik
Hintergrundwissen	Speditionen, Technikbegriffsverständnis kann nicht vorausgesetzt werden	Ehemaliger Beruf, Technikbegriffsverständnis kann nicht vorausgesetzt werden
Interaktionsfähigkeiten		
Aufmerksamkeitsspanne	beim Autofahren extrem kurz	siehe Benutzertyp 1
Motorische Fähigkeiten	nur rechte Hand, ein Finger, der ein Gerät an einer Halterung bedient	siehe Benutzertyp 1
Audiovisuelle Fähigkeiten	Während der Fahrt Konzentration auf Verkehr, blinde Interaktion;	siehe Benutzertyp 1, aber eher Brillenträger, Hörgeräte mit steigendem Alter
Mentale Fähigkeiten	Multitasking	siehe Benutzertyp 1
Benutzungsverhalten		
Bevorzugter Ort der Benutzung	Autobahnen	Autobahnen
User type (aktiv/passiv konsumierend)	aktiv	aktiv
Multimedia und Anwendungsnutzung	wird nicht vorausgesetzt	wird nicht vorausgesetzt

Tabelle 33: Hauptzielgruppen des TrafficScanners: Berufsautofahrer und Rentner, die viel und/oder gerne fahren

Mittels der im Papierprototyp abgebildeten Ansichten der Anwendung wurde das Hauptszenario durchgespielt; häufig gestellte Fragen der Benutzer wurden als Probleme zusammengefasst. Mit einem Brainstorming wurden dann Hauptaufgaben der Anwendung und einzelne Interaktionen aufgelistet:

- Programm installieren und deinstallieren
- Installieren von externen Positionierungssystemen mithilfe von Bluetooth
- Verwenden von externen Positionierungssystemen mithilfe von Bluetooth
- Programm starten
- Programm beenden
- Zwischen dem Programm und anderen Programmen auf der Plattform wechseln
- Anmelden an das System
- GPS- bzw. Netz-Verbindung an- und ausschalten
- Staumeldung (Hauptszenario)
  - Anmelden
  - Staubeginn melden
  - Ende des Staus melden
  - Abmelden
- Übersicht über eigene Meldungen erhalten
  - Erfolgskontrolle
  - Kostenkontrolle (falls Meldungen nicht kostenfrei)
  - Anzahl der Nachrichten, der Verbindungen

Ergänzt wurden die Anforderungen, dass angezeigt werden soll, dass der Server kontaktiert wird und die Benutzer deshalb warten müssen, die Fehlerbehandlung der Anmeldung, das Abbrechen und Wiederholen eines Sendevorgangs, das Logging von Daten, der Stromverbrauch durch GPS und die Integration von externen GPS-Geräten.

Hinsichtlich der mobilen Endgeräte unterstützt die analysierte Anwendung nur folgende mobile Endgerätetypen mit den mobilen Plattformen Symbian (Endgerätetyp 1) und Windows Mobile (Endgerätetyp 2):

- Displaygrößen (Breite x Höhe): 240 x 320 Pixel, 320 x 240 Pixel
- Betriebssysteme: Windows Mobile 5 PocketPC, Windows Mobile 6 Professional, bestimmte Endgeräte von Nokia und Samsung mit Symbian S60
- Touchscreen: erforderlich (eine Bedienung auf Smartphones ist nicht vorgesehen)
- Telefon-Modul: CSD-fähig (Modemeinwahl)
- GPS-Modul: integriert oder über Bluetooth

5.3.3.3 *Mapping von Interaktionsklassen auf Eigenschaften des Nutzungskontexts*

Die resultierende Liste von Interaktionsklassen und Qualitätsmerkmalen wird in Tabelle 34 aufgeführt.

Anforderungen aus Nutzungskontextfaktoren	*Interaktionskategorie*
Autofahren verbietet Handynutzung (B1-B5) Auto (U1) Beifahrer (U3) Eingabe via Touchscreen (EG5) Stromverbrauch GPS, Möglichkeit der automatischen Abschaltung (EG7) Anmelden (A1)	1. Erfassung und Behandlung von Benutzereingaben
Plattformen: Konformität beachten (EG6)	3. Navigation, Browsen, Auswählen
Netzabdeckung (MA8) Senden (A1)	4. Ausführen, Wiederholen und Zurücknehmen von Befehlen
Auto, gesetzliche Rahmenbedingung (U1), Muss-Anforderung (WK3) Kollaboration mit externen GPS-Empfängern (U5), Muss-Anforderung (WK1)	5. Interagieren mit unbekannten Systemen
Senden (A1)	8. Austausch und Manipulation von Daten
Plattformen: Konformität beachten (EG6)	10. Strukturieren und Anzeigen von Content, Information oder Daten
Netzabdeckung (MA8 und U4) Fehler durch Ablenkung durch Beifahrer (U3) Senden wiederholen ermöglichen (A1)	11. Behandlung von Fehlern und Hilfe
Kontextänderung führt zu systeminitiierter Interaktion (WK1) nur ein Klick (WK3)	12. Adaptation (durch Benutzer, für Benutzer, für Aufgaben)
externe GPS-Empfänger (MA1, U5) Netzabdeckung (MA8, U4)	13. Kooperation (in einem System, mit anderen Systemen)

Tabelle 34: Beispiel-Mapping von Nutzungskontext und Interaktionskategorien der Fallstudie TrafficScanner

5.3.3.4 *Beschreiben der Softwarearchitektur*

Die zur Analyse vorhandene Dokumentation wurde bereitgestellt und durch Informationen auf der Webseite des Projekts ergänzt. Es handelte sich um die Präsentation der Benutzerschnittstelle (Papierprototyp) und die Feinspezifikation. Es lagen häufig gestellte Fragen an den Support und zwei Benutzerhandbücher vor, je eins für Symbian S60 und Windows Mobile. Weder der Quelltext noch eine laufende Version der Software waren vorhanden.

Die Projektdokumentation umfasste den Papierprototypen und die Spezifikation. Diese enthielt Personas, User Stories und Dialoge in Form von Szenarios mit EPK-Diagrammen (EPK: Event-Prozess-Kette) und einer für Analysezwecke zu allgemeine Architekturbeschreibung. Es wurde daher zusätzlich eine neu generierte Doku-

mentation (Doxygen) bereitgestellt, allerdings ohne UML-Diagramme. Da es keine detaillierte Softwarearchitekturbeschreibung gab, wurde sie punktuell, d.h. soweit wie zum Verständnis nötig, selbst erstellt. Ein Entwickler stand während der Analyse als Ansprechpartner per Instant Messaging zur Verfügung.

Klassifikation	Ausprägung
Technische Rahmenbedingungen	
Hardware-Vorgaben	siehe Nutzungskontext Mobile Endgeräte
Software-Vorgaben	Plattformen, Client-Server
Vorgaben Systembetrieb	Windows Mobile 5 PocketPC, Windows Mobile 6 Professional, Symbian S60
Programmiervorgaben	C++
Organisatorische Rahmenbedingungen	
Organisation und Struktur	Prozess: Extreme Programming
Ressourcen (Budget, Zeit, Personal)	k.A.
Organisatorische Standards	k.A.
Juristische Standards	Einfingerbedienung im Automobil – Straßenverkehrsordnung
Konventionen	k.A.
Kontextabgrenzungen	
Fachlicher Kontext	Geschwindigkeit des Automobils
Technischer- oder Verteilungskontext	GPS-Daten, Batterielaufzeit, Verfügbarkeit des Netzes, Verfügbarkeit der GPS-Daten

Abbildung 44: Rahmenbedingungen von TrafficScanner

Die aufwendigste Tätigkeit während der ersten Phase war deshalb, anfangs die Struktur der Softwarearchitektur zu erstellen (siehe Abbildung 45) und alle Schnittstellen und ihre Aufgaben zu beschreiben[4]. Die generierte Dokumentation war die wichtigste Informationsquelle, da alle Elemente und Schnittstellen im Detail anhand des Quelltexts automatisch dokumentiert waren.

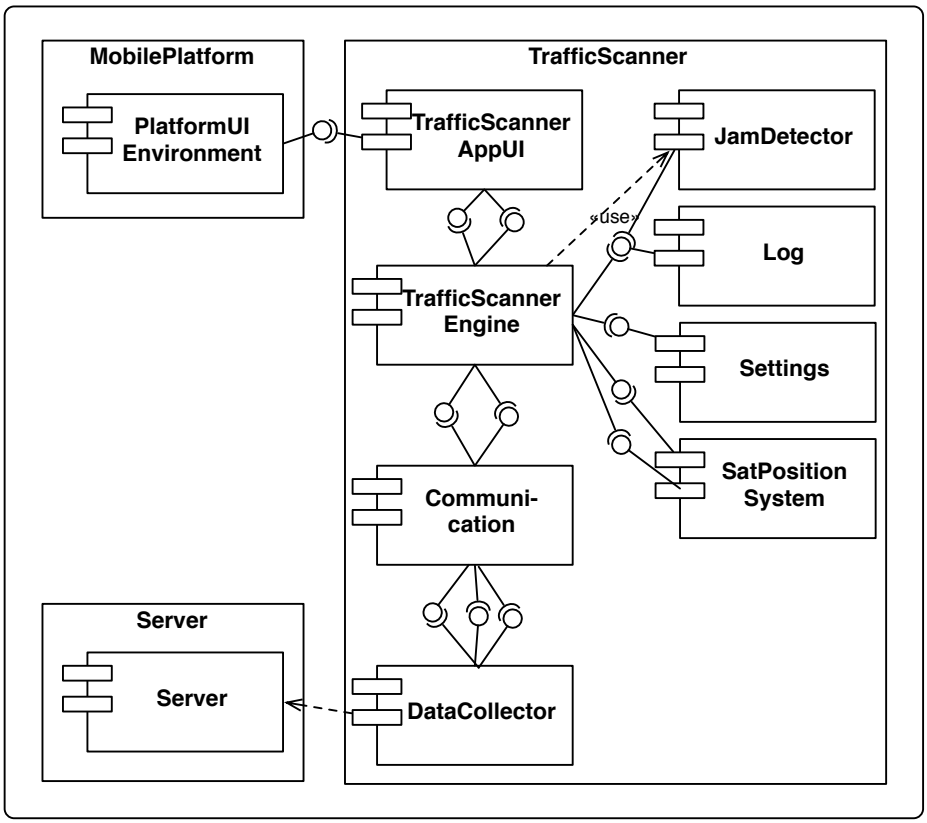

Abbildung 45: TrafficScanner-Softwarearchitektur im Überblick

4 Die Beschreibung der Softwarearchitektur ist in SATURN jetzt eine Voraussetzung und ggf. eine im Vorfeld durchzuführende Aktivität.

Die Anwendung besteht aus sieben Komponenten, mit folgenden Verantwortlichkeiten:

- `TSAppUI`: enthält alle Masken und Bedienelemente,
- `Settings`: ermöglicht die Konfiguration der Anwendung (Schwellwerte, Logging, etc.), enthält auch Grenzgeschwindigkeiten für Autobahn (über 120 km/h), Stau (unter 30 km/h) und Autobahnabfahrt (unter 50 km/h),
- `JamDetector`: wertet die Positionsdaten aus,
- `Logging`: es können je nach konfiguriertem Detailgrad Informationen zum Nutzer-/Anwendungsverhalten zu Diagnosezwecken geloggt werden (0 – Aus [Default] Logging ist ausgeschaltet und 5 – Trace),
- `SatPositionSystem`: stellt die GPS-Daten zur Verfügung,
- `TSEngine`: Zentrale Komponente, überprüft die Schwellwerte, Benutzereingaben und steuert den Anwendungsablauf,
- `Communikation`: stellt die Verbindung zum Kommunikationsserver für den Meldungsversand her und abstrahiert den Übertragungsweg (für zukünftige Erweiterung auf GPRS oder SMS),
- `DataCollector`: übermittelt die Daten zum Server.

Schnittstelle	Komponente/Nachbarschaftssystem
GPS	• GPS-Komponente: Stellt die GPS-Daten zur Verfügung • Interface: 6 – GpsInterface. Benutzt durch TSEngine, Übermittlung via 7 – GpsCallBack • Aufgabe: GPS Positionsevents liefern • Format GpsEvent, Medium: Intern; Extern: Bluetooth (vom OS bereitgestellte Positionsübermittlung wird verwendet unabh. von Art der Ortsbestimmung, dh. GPS o.a.) • Systemerweiterung ist Aufgabe der Platform-Implementierung: Kooperation ermöglichen, z.B. bessere Positionsdatenübermittlung, Intergration mit Navigationssystemen (Navis) • GPS-Fehler sind schwierig händelbar (Verbindungsfehler evtl. nicht identifizierbar) • Implementiert durch Objekte einer Klasse Gps
Logging	• Logging-Komponente: Es können je nach konfiguriertem Detailgrad Informationen zum Nutzer-/Anwendungsverhalten zu Diagnosezwecken geloggt werden. • 4 – TSLoggin: Wird benutzt von TSEngine und JamDetector • Hinweis: auch TSEngine nutzt den JamDetector • Aufgabe: Loggt und schreibt in Logging File auf dem Endgerät, es gibt 5 Log-Levels, höchster Level: trace. • Implementiert durch Log (separate Klasse), gehört zur TSEngine (im Sinne von Speichermanagement, Owner muss zerstören z.B.) • JamDetector nutzt Settings, ebenso das letzte GPS Ereignis, speichert 5 GPS Ereignisse insgesamt, ermittelt Geschwindigkeit, meldet Stau
Settings	• Konfigurations-Komponente: Ermöglicht die Konfiguration der Anwendung (Schwellwerte, Logging, etc.) • 5 – Setttings, wird benutzt von TSEngine • Aufgabe: nur Konfigurationsstellungen speichern (km/h-Schwellwerte für die Trigger, Logging) • Benutzt Textinifile, implementiert durch Setting • Information wird gesendet via TSEngine in Kooperation mit Communication
UI Events verarbeitend	• 1 – UIEventandler • Wird implementiert von TSEngine: UI hart einkodiert • Vorgaben fürs UI sind nicht platformkonform, S60 kann nur Listen, normal hoch und doppelthoch, Fullscreen Extraarbeit, extra Framework. • Benutzt durch TSAppUI
Schnittstelle zum UI	• UI-Komponente: Enthält alle notwendigen Masken und Bedienelemente • nicht im Diagramm enthalten: Interface ScreenCommandOberserver • Implementiert von TSAppUI, gibt programmrelevante Events weiter an TSEngine • Wird benutzt von Objekten von ScreenController (Basisklasse für alle Screens)
TSEngine	• Logik-Komponente, überprüft Schwellwerte, Benutzereingaben, steuert Anwendungsablauf • zentrale Logik für die Verarbeitung
Kommunikationsinterface	• Kommunikations-Komponente stellt die Verbindung zum Kommunikationsserver für den Meldungsversand her und abstrahiert den Übertragungsweg (für Erweiterung auf GPRS oder SMS) • 9 – CommInterface: benutzt durch TSEngine, implementiert durch Communication • Aufgabe: Nachrichten zum Server schicken • Medium: Luft. Protokoll: UDP; • Symbian/Windows: CSD, sobald neue Abrechnungsmodelle mit Kostenübernahme auch andere • Modularer Aufbau: Komponente leicht austauschbar • Callback via 8 – CommCallBack • Aufgabe: Fehlermeldungen, Konfigurationsinformationen von Server weiterleiten
Interne Interfaces in der Kommunikationsschnittstelle	• ChannelObserver für internes Callback; ConnectionObserver ebenso • Aufgabe: technische Implementierung CSD, Notifikation über Async Request(Daten erhalten/versendet) • Beide implementiert von Communicator • Benutzt von DataCollector (Implementierung des Channels für CSD, kommuniziert mit Server) • ComChannel, ComChannelObserver gehören zusammen • Communication liefert Daten via ComChannel, werden durch CSD verschickt, via ConnectionObserver Bericht zum Status der Verbindung; ComChannelOberser berichtet „Fertig“.

* alle originalen Interfaces, Klassen, Komponenten wurden umbenannt

Abbildung 46: Schnittstellenbeschreibung mit Schnittstellen zwischen Klassen/Komponenten sowie Format und Medium

Der Kontext der Systeminteraktion ist definiert als Geschwindigkeit, nicht als Ort (auch nicht Bewegungsdaten des mobilen Endgerätes). Bei Änderung aktiviert ein Event, dass eine Meldung durch das User Interface angezeigt wird.

Nachdem diese inhaltlichen Themen bearbeitet wurden, wurden abschließend die Teilnehmer für die Phase 2 festgelegt. Es wurden keine weiteren Teilnehmer (wie beispielsweise Stakeholder) für die zweite Phase eingeladen.

5.3.4 *Durchführung der Fallstudie TS: SATURN-Phase 2*

5.3.4.1 *Vorauswahl von Interaktionsszenarios aus CASSINI*

Da die Phase 1 nicht am gleichen Tag durchgeführt wurde, wurden anfangs in 20 Minuten die Ergebnisse der Phase 1 präsentiert und zusammengefasst.

Danach wählten die Teilnehmer Interaktionsszenarios aus dem Katalog aus. Die zwei Analysten und der Softwarearchitekt prüften jeweils allein, welche der Interaktionsszenarios ihrer Meinung nach für die aktuelle Software relevant sein könnten. Die Begründungen wurden notiert.

5.3.4.2 *Diskussion und Bestimmen der zu evaluierenden Interaktionsszenarios*

Bei der Diskussion der einzelnen Interaktionsszenarios äußerte sich der Architekt immer zuerst. Es wurde deutlich, dass er alle Interaktionsszenarios so verstanden hatte, wie sie gemeint waren. Er wählte Interaktionsszenarios aus, die aus seiner Sicht mit einer wichtigen Usability-Anforderung zusammenhängen. Der Analyst wählte Interaktionsszenarios aus, die Interaktionen von Phase 1 betreffen und setzte deren Auswahl durch. Die Abbildungen 47 und 48 dokumentieren die Auswahl der Interaktionsszenarios.

Szenario		Einzelurteile: Szenario zutreffend?			Endgültiges Urteil	Kommentar (bei uneinheitlichen Einzelurteilen)
ID	**Name**	**AR**	**AN1**	**AN2**		
1	Observing System State	Ja	Ja	Ja	Ja	
2	Multiple Data Selection	Nein	Nein	Nein	Nein	
3	Macros	Nein	Nein	Nein	Nein	
4	Canceling Commands	Nein	Vielleicht	Ja	Ja	AR hielt Funktion zum Abbruch der Datenübertragung an den Server für unnötig; vermutet wird aber, dass gleichzeitiges Telefonieren und Uploaden technisch nicht möglich
5	Forgiving Format	Ja	Ja	Ja	Ja	
11	Good Defaults	Nein	Nein	Nein	Nein	
12	Auto-Completion	Nein	Nein	Nein	Nein	
14	Checking for Correctness	Ja	Ja	Ja	Ja	
17	Command History	Vielleicht	Ja	Nein	Nein	Übersicht bislang versandter Meldungen wurde abgelehnt, da Anwendung schlicht gehalten, Nutzen fraglich, von Spezifikation nicht gefordert; vorbehaltlich anderslautender Auskunft des Auftraggebers (Rückfrage)
18	Alternative Views	Nein	Nein	Nein	Nein	
19	User Modes and Profiles	Nein	Nein	Nein	Nein	
20	Supporting International Use	Nein	Vielleicht	Vielleicht	Nein	Keine fremdsprachigen Versionen geplant; evtl. für den inländischen Gebrauch dennoch gewünscht, z. B. auf Türkisch, Englisch (Rückfrage an Auftraggeber)
21	Multi-Channel Access	Nein	Vielleicht	Nein	Nein	Unterstützung optionaler, von Kernfunktionen des Systems unabhängiger Geräte derzeit nicht vorgesehen
23	Multi-Level Undo	Nein	Nein	Nein	Nein	
24	Workflow Model	Ja	Ja	Nein	Ja	Modifikation: Optionen nicht an Workflow, sondern an Kontext ausrichten, somit geringe Abweichung von Szenario 24
25	Using Applications Concurrently	Ja	Ja	Ja	Ja	
26	Maintaining Device Independence	Vielleicht	Ja	Ja	Ja	Anschluss von unterstützten (GPS-Empfänger) oder nicht unterstützten Geräten an das Handy darf ordnungsgemäßen Betrieb nicht beeinträchtigen
28	Recovering from Failure	Nein	Nein	Nein	Nein	
29	Retrieving Forgotten Passwords	Nein	Nein	Nein	Nein	
30	Reusing Information	Nein	Nein	Nein	Nein	
31	Paste Variations	Nein	Nein	Nein	Nein	
32	Supporting Multiple Activities	Nein	Nein	Nein	Nein	
34	Working at the User's Pace	Nein	Nein	Nein	Nein	
35	Progress Indication	Ja	Ja	Ja	Ja	
36	Comprehensive Searching	Nein	Nein	Nein	Nein	

Abbildung 47: Ergebnis der Szenario-Auswahl, Seite 1

37	Familiar Appearance and/or Behaviour	Ja	Nein	Ja	Ja	Softkey-Belegung soll mit Plattform-Standard übereinstimmen; auf diese Begründung eigentlich besser passend: Szenario Nr. 127 (Standard Softkey Behaviour; wg. Architektur-Ratings nicht unter den 42 Versuchs-Szenarien)
39	New or Traditional Interface	Nein	Nein	Nein	Nein	
40	Working in an Unfamiliar Context	Nein	Nein	Nein	Nein	
41	Verifying Resources	Ja	Ja	Ja	Ja	
42	Operating Consistently Across Views	Nein	Nein	Nein	Nein	Beispiel erbeten
44	Consistent Set of Views	Nein	Nein	Nein	Nein	
49	Navigating within a Single View	Nein	Nein	Nein	Nein	
51	Location Information	Nein	Ja	Nein	Nein	Szenario regelt einheitliche Behandlung von Adressdaten und nicht, wie von AR, AN1 und AN2 zunächst angenommen, die Ermittlung des aktuellen geografischen Aufenthaltsortes; selbst diese wird jedoch verneint unter Verweis auf die Spezifikation; unzutreffendes Architektur-Rating
56	Error Messages	Ja	Ja	Ja	Ja	
62	Visual Browsing	Nein	Nein	Nein	Nein	
72	Wizard	Nein	Nein	Nein	Nein	
76	Help	Ja	Ja	Ja	Ja	
79	Direct Access to Sections	Nein	Nein	Nein	Nein	
83	Breadcrumbs	Nein	Nein	Nein	Nein	
87	Overview and Access to Sections	Nein	Nein	Nein	Nein	
102	Smart Menu Items	Nein	Ja	Nein	Nein	Folgen mancher Optionen unklar, jedoch nicht weil Option erklärungsbedürftig, sondern weil Spezifikation lückenhaft; soweit nach Auswahl einer Option weitere Navigation unklar, sollte nicht die Erklärung, sondern der Zustand nach Aufruf der Option verbessert werden
106	Dynamic Queries	Nein	Vielleicht	Nein	Nein	Urteil von AN1 bezog sich tatsächlich auf Szenario Nr. 12 (Auto-Completion)

Abbildung 48: Ergebnis der Szenario-Auswahl, Seite 2

Zwölf Interaktionsszenarios wurden ausgewählt. Vier davon begründen sich auf die Ergebnisse der ersten Phase. Bis auf eine Ausnahme wurde kein Interaktionsszenario inhaltlich angepasst: ein neues Interaktionsszenario wurde erstellt, indem ein vorhandenes Interaktionsszenario modifiziert wurde. Einige beim Walkthrough in Phase 1 entdeckten Interaktionen wurden von der Analyse ausgeschlossen, da ihre Relevanz erst mit Hilfe einer User-Evaluation überprüft werden sollte. Die Abbildung 49 listet auf, welche Interaktionsszenarios erweitert wurden.

Szenarios	Begründung	Sensitive Entscheidungen	Unterstützung des Szenarios	Max. Einfluss auf Usability, Softwarearchitektur (ASL aktuell)	ASL0 (erste Analyse)	ASL1 (folgende Iteration)	ASL 2 (folgende Iteration)
Observing System State (System Feedback)	Walkthrough						
Abbrechen (Canceling Commands)	Kontext: Walkthrough						
Forgiving Format	Walkthrough						
Checking for Correctness	Walkthrough						
Context-based Workflow	Hauptszenario						
Using Applications Concurrently	Katalog						
Maintaining Device Independence	Katalog						
Progress Indication	Katalog						
Familiar Appearance and/or Behaviour	Katalog						
Verifying Resources	Katalog						
Error Messages	Katalog						
Help	Katalog						

Einfluss auf Usability
0... Die Anforderung wird umgesetzt.
1... Die Anforderung wird teilweise umgesetzt.
2... Die Anforderung wird nicht umgesetzt.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.

Einfluss auf Softwarearchitektur
0... Keine Modifikationen werden erwartet.
1... Einfache Modifikationen werden erwartet.
2... Komplexe Modifikationen werden erwartet.
3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

* abh. von Abrechnungsmodellen der Provider

Abbildung 49: ASL-Tabelle für die Analyse von TrafficScanner

5.3.5 *Durchführung der Fallstudie TS: SATURN-Phase 3*

Die folgenden Tabellen zeigen die originalen Analyseformulare. Da die Analyse in englischer Sprache und vor der Überarbeitung des Kataloges CASSINI erfolgte, werden die neuen und die deutschen Namen der Interaktionsszenarios nur hier genannt.

- Observing System State (System Feedback)
- Canceling Commands (Abbrechen, Cancel)
- Forgiving Format (Nachsichtiges Formular)
- Checking for Correctness (Prüfen auf Korrektheit)
- Workflow Model (Workflow-Modell)
- Using Applications Concurrently (Konfliktfreie Nebenläufigkeit, Non-conflicting Application Concurrency)
- Maintaining Device Independence (Geräteunabhängigkeit, Device Independence)
- Progress Indication (Fortschrittsanzeige)
- Familiar Appearance and/or Behaviour (Vertrautes Aussehen und Verhalten)

- Verifying Resources (Prüfen notwendiger Ressourcen)
- Error Message (Fehlermeldungen)
- Help (Mehrstufige Hilfe)

5.3.5.1 *System-Feedback*

Observing System State (System Feedback)

Allgemeine Informationen Nummer: 1 Kurzname: Feedback Anforderungskategorie: Intention Interaktionskategorie(n): Orientation, Structuring and Displaying Content/Information/Data, Adaptation (by Users/to Users/for Tasks)	**In die Analyse aufgenommen via (zutreffendes unterstreichen)** Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Rating mit Begründung** Architekt: ja Analyst 1: ja Analyst 2: ja Katalog: ●●●● High (based on literature)
Szenario Umwelt: Automobil Initiator: „Users“ (Administrator) Stimulus: „want to stay informed about the system's state, activities and changes.“ Umgebung: Mobiles Endgerät Response: „The system picks the data required by the user and presents them according to human needs and capabilities. “ Response Measure: „Users are supplied with all necessary data.U Users perceive all necessary data. (U) The data perceived by the users are up to date. (U)“	**Rationale, Referenzen** Usability-Attribute: Efficiency, Safety, Utility, Learnability Usability-Anforderung: Benutzer müssen eine Reaktion auf ihre Aktion bemerken und wissen, was das System gerade macht HCI-Patterns: keine. SE-Rational: Observing System State [BJK01], System Feedback [Fol05] Projektreferenzen: keine.

Analyse
Use Case(s): Staumeldung (Hauptszenario).

Nr	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
1	Feedback	Staumeldung (Hauptszenario)	`TSEngine`, `Communicator`	12	keins	Nichtexistenz eines Status "ich versende gerade" für Feedback oder eine „Meldungswarteschlange“

Sensitive Entscheidungen: 1 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
1	Nichtexistenz eines Status "ich versende gerade" für Feedback oder eine „Meldungswarteschlange“	User sehen nicht, ob Meldung noch in Warteschlange ist. Sie sollten es aber sehen., Feedback über Warteschlange → Response nicht unterstützt	2	`TSAppUI`, `TSEngine`, `Comm.` und `Interface` sollten angepasst werden;	2	[-] Fehlertoleranz	ja

Zusammenfassung
Usability-Problem: User sehen nicht, ob Meldung noch in Warteschlange ist, ob sie gesendet wird oder noch gesendet werden. Feedback über Warteschlange; Response nicht unterstützt → 2

SA-Sensitivität: `TSAppUI`, `TSEngine`, `Communicator` und `Interface` sollten angepasst werden; Meldungswarteschlange vorgeschlagen → 2

Tradeoffs: [-] Fehlertoleranz bei Netzausfall oder bei Abbruch des Sendens
Andere(s) Szenario(s): Progress Indicator
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,2)
Ziele: ASL 1. Iteration: (0,0) – höchste Priorität, ASL 2. Iteration: (0,0)

Usability-Evaluation: Testen des Hauptszenarios.
Sonstige Kommentare: keine

Abbildung 50: Analyseformular System-Feedback

5.3.5.2 *Abbrechen*

Abbrechen (Canceling Commands)

Allgemeine Informationen Nummer: 4 Kurzname: Abbrechen (Cancel) Anforderungskategorie: Robustheit Interaktionskategorie: Executing, Repeating and Revoking Commands	**In die Analyse aufgenommen via** Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Begründung der Auswahl** Architekt: nein (ist Funktion für Abbruch des Sendens nötig?) Analyst 1: vielleicht Analyst 2: ja Katalog: ●●●● High (based on literature)
Szenario Umwelt: Automobil Initiator: „Users“ Stimulus: „started an operation but now don't want it to be executed any longer“ Umgebung: Mobiles Endgerät Response: „The system immediately stops execution as users activate the cancelling option.“ Response Measure: „Cancellation is performed and communicated to the user within a certain time span (A). System state before starting the operation is restored. (A)“	**Rationale, Referenzen** Usability-Attribute: Effectiveness, Efficiency, Safety, Learnability HCI-Rational: Kontrolle über das System, Benutzungs- oder Systemfehler ohne große Auswirkungen HCI-Patterns: *Canceling Commands* [BJK01], *Cancelability* [Tid06] SE-Rational: An operation cannot cancel itself. There have to be made monitoring and executing provisions to achieve a robust cancel interaction. SE-Patterns: [Fol05] *Cancel*: Provision for the componetns monitoring user input, this should be independent, concurrent; components processing actions can be interrupted, consequences of actions are rolled back. Projektreferenzen: bisher keine.

Analyse
Use Case(s): Abbrechen des Sendens, um beispielsweise einen eiligen Anruf zu tätigen oder um die Batterie zu schonen

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
2	Cancel	Abbrechen des Sendens	`TSEngine`, `TSAppUI`, `Communicator`	zum System	keine	Cancel des Sendens wird architekturell nicht unterstützt

Sensitive Entscheidungen: 2 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Trade-offs +/-	Kritisch
2	Cancel des Sendens wird architekturell nicht unterstützt	Abbruch des Sendens wird architekturell nicht unterstützt → Response nicht unterstützt	2	Änderungen in mehreren Komponenten nötig, evtl. Cancel-Komponente, Koop. mit Plattform ist möglich (CActive-Framework) wird aber aktuell nicht unterstützt	2	k.A.	ja

Zusammenfassung
Usability-Problem: Abbruch des Sendens wird architekturell nicht unterstützt. Response nicht unterstützt. → 2

SA-Sensitivität: Änderungen in mehreren Komponenten nötig, evtl. Cancel-Komponente, Koop. mit Plattform ist möglich (CActive-Framework) wird aber aktuell nicht unterstützt → 2.

Tradeoffs: k.A.
Andere(s) Szenario(s): Feedback
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,2)
Ziele: ASL 1. Iteration: (0,0), ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Mit nächster Version: Beobachten und bewerten, wie die Benutzer die Verbindung abbrechen.

Abbildung 51: Analyseformular Abbrechen

5.3.5.3 *Forgiving Format*

Forgiving Format

Allgemeine Informationen
Nummer: 5
Kurzname: Forgiving Format
Anforderungskategorie: Robustheit
Interaktionskategorie: Acquiring and Processing User Input

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●○ High

Szenario
Umwelt: Automobil
Initiator: „System“
Stimulus: „prompts users for data using a form“
Umgebung: Mobiles Endgerät
Response: „Users enter the data into a single field and do not need to respect a specific format or syntax.“
Response Measure: „Where logically possible, the system recognizes kind and format of input data correctly. (U) “

Rationale, Referenzen
Usability-Attribute: Efficiency, Safety, Utility, Learnability
HCI-Rational: Kontrolle über das System, Benutzungs- oder Systemfehler ohne große Auswirkungen
HCI-Patterns: *Forgiving Format* Permit users to enter text in a variety of formats and syntaxes, and make the application interpret it intelligently. [Tid06]
SE-Rational: Fault-Tolerance
SE-Patterns: k.A.
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Login

Nr	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
3	Forgiving Format	Login	`TSAppUI, TSEngine,`	2, 3	keins	Feedback bei Fehler: Eingabe der Staumeldernummer unter 6 Stellen mit führender Null

Sensitive Entscheidungen: 3 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
3	Feedback bei Fehler: Eingabe der Staumeldernummer unter 6 Stellen mit führender Null	Eingabefehler wird nicht automatisch korrigiert, oder angezeigt → Response wird nicht unterstützt	2	Änderung der Behandlung der Eingabedaten in den bestehenden Komponenten. Ergänzen von Dialog und entsprechenden Funktionen.	1	keine	ja

Zusammenfassung
Usability-Problem: Eingabefehler wird nicht automatisch korrigiert, oder angezeigt. Response wird nicht unterstützt. → 2

SA-Sensitivität: Änderung der Behandlung der Eingabedaten in den bestehenden Komponenten. Ergänzen von Dialog und entsprechenden Funktionen. → 1.

Tradeoffs: k.A.
Andere(s) Szenario(s): Checking for Correctness
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,1)
Ziele: ASL 1. Iteration: (2,1); ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Beobachten und bewerten, inwiefern die Benutzer mit dem Fehler zurecht kommen.
Sonstige Kommentare: keine.

Abbildung 52: Analyseformular Forgiving Format

5.3.5.4 *Checking for Correctness*

Checking for Correctness

Allgemeine Informationen
Nummer: 14
Kurzname: Checking for Correctness
Anforderungskategorie: Robustheit
Interaktionskategorie: Acquiring and Processing User Input, Error Handling and Help

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „System“
Stimulus: „wants to ensure the data users have entered are valid and correct“
Umgebung: Mobiles Endgerät
Response: „Once the users submit the form, the system checkst he input for errors and – if there are any – starts an error handlich routine.“
Response Measure: „Input errors are detected by the system (A). Input errors detected by the system are handled (A).“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Efficiency, Safety, Learnability
HCI-Rational: Kontrolle über das System, Benutzungs- oder Systemfehler ohne große Auswirkungen
HCI-Patterns: [BJK01] 5 – *Checking for Correctness*: A user may make an error that he or she does not notice. Systems should suggest or perform error correction.
[Fo05] 20 – *Data Validation*: Verify whether (multiple) items of data in a form or field have been entered correctly.
SE-Rational: k.A.
SE-Patterns: k.A.
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Login

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
4	Checking for Corr.	Login	`TSEngine, TSAppUI`	1	keins	Speicherung der Login-Daten (eine Zahl) als Integer

Sensitive Entscheidungen: 4 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
4	Speicherung der Login-Daten (eine Zahl) als Integer	Eingabe wird auf 6-stellige Nummer geprüft, falls Login 5-stellig muss laut FAQ vorangehende Null ergänzt werden. Response wird umgesetzt, aber falsch. Integer-Werte speichern vorangehende Nullen nicht; Benutzerfehler und Mehrfacheingabe wird provoziert → Response wird teilweise unterstützt	1	Änderung in TSAppUI	1	keine	ja

Zusammenfassung
Usability-Problem: Eingabe wird auf 6-stellige Nummer geprüft, falls Login 5-stellig muss laut FAQ vorangehende Null ergänzt werden. Response wird umgesetzt, aber falsch. Integer-Werte speichern vorangehende Nullen nicht; Benutzerfehler und Mehrfacheingabe wird provoziert: Response wird teilweise unterstützt → 1

SA-Sensitivität: Änderung der Eingabekontrolle, so dass gleich auch kleinere Integer-Werte verwendet werden → Änderungsaufwand ist 1.

Tradeoffs: keine
Andere(s) Szenario(s): Forgiving Format
Architekturdiagramm(e): k.A.

Abbildung 53: Analyseformular Checking for Correctness, Seite 1

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (1,1)
Ziele: ASL 1. Iteration: (1,1), ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: siehe Forgiving Format.
Sonstige Kommentare: keine.

Abbildung 54: Analyseformular Checking for Correctness, Seite 2

5.3.5.5 *Context-based Workflow*

Context-based Workflow

Allgemeine Informationen
Nummer: 24
Kurzname: Workflow Model **(modifiziert)**
Anforderungskategorie: Intuition
Interaktionskategorie: Data Exchange and Manipulation, Adaptation (by Users/to Users/for Tasks)

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes -- Hauptszenario

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: nein
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „Users“
Stimulus: „perform, as a single step based on context changes, a specific task on a piece of data.“ **(modifiziert)**
Umgebung: Mobiles Endgerät
Response: „Exactly the tools/actions are provided that users need to perform their specific task.“
Response Measure: „Users carry out their task by means of the provided tools and actions. (U) All of the provided tools and actions have to be available to make sure the task can be carried out. (A)“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Efficiency, Safety, Utility
HCI-Rational:
HCI-Patterns: [Fo05] 29 – *Workflow Model*: Provide different users only the tools or actions that they need in order to perform their specific task on a piece of data before passing it to the next person in the workflow chain.
SE-Rational:
SE-Patterns:
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Staumeldung

Nr	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
5	Context-based Workflow	Staumeldung (Hauptinteraktion)	`TSEngine`, `TSJam-Detector`	4	keins	System fordert Benutzereingabe basierend auf aktueller Geschwindigkeit
6	Feedback	Staumeldung (Hauptszenario)	TSAppUI, TSEngine	8	keins	Ablauflogik bei Staumeldung: aktuelle Daten werden verwendet, statt die, die zur Stauerkennung geführt haben

Sensitive Entscheidungen: 5 (gut), 6 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
5	System fordert Benutzereingabe basierend auf aktueller Geschwindigkeit	Kein negativer Einfluss. Response wird unterstützt.	0	Keine.	0	keine	nein
6	Ablauflogik bei Staumeldung: aktuelle Daten werden verwendet, statt die, die zur Stauerkennung geführt haben	Fehler provoziert, falls keine GPS-Daten in dem Moment vorliegen Stauerkennung-User soll bestätigen - dann kann es doch nicht gesendet werden: doppelte Arbeit nötig, Erwartungen nicht erfüllt. Staumeldung auch doppelt falls kein Empfang. Workflow wird also im Fehlerfall nicht beendet → Response nicht unterstützt	2	`TSEngine`, `JamDetector` müßten Daten speichern, die zur Stauerkennung geführt haben.	1	[-] Fehlertoleranz des Systems	ja

Abbildung 55: Analyseformular Context-based Workflow, Seite 1

Zusammenfassung
Usability-Problem: Fehler provoziert, falls keine GPS-Daten in dem Moment vorliegen
Stauerkennung-User soll bestätigen - dann kann es doch nicht gesendet werden: doppelte Arbeit nötig, Erwartungen nicht erfüllt. Staumeldung auch doppelt falls kein Empfang. Workflow wird also im Fehlerfall nicht beendet; Response nicht unterstützt
→ 0

SA-Sensitivität: `TSEngine`, `JamDetector` müßten Daten speichern, die zur Stauerkennung geführt haben. → 1

Tradeoffs: [-] Fehlertoleranz des Systems
Andere(s) Szenario(s): Feedback
Architekturdiagramm(e): k.A.

Fazit: Szenario wird teilweise unterstützt (Problem bei GPS-Ausfall).
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,1)
Ziele: ASL 1. Iteration: (0,0); ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Testen des Hauptszenarios
Sonstige Kommentare: keine.

Abbildung 56: Analyseformular Context-based Workflow, Seite 2

5.3.5.6 *Using Applications Concurrently*

Using Applications Concurrently

Allgemeine Informationen
Nummer: 25
Kurzname: Non-conflicting Application Concurrency
Anforderungskategorie: Robustheit
Interaktionskategorie: Exchange/Cooperation Within and Between Systems

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „Users“
Stimulus: „want to work with multiple applications at the same time.“
Umgebung: Mac/PC
Response: „The system allows for being run concurrently to other software without conflicts.“
Response Measure: „While applications are running concurrently, no errors occur. (A)“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Efficiency, Safety
HCI-Rational:
HCI-Patterns: [BJK01] 4 – *Using Applications Concurrently*: A user may want to work with arbitrary combinations of applications concurrently. Systems should ensure that users can employ multiple applications concurrently without conflict.
SE-Rational:
SE-Patterns:
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Parallelbetrieb Telefon

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
7	Using Appl. Conc.	Parallelbetrieb Telefon	`Communication`	10, 11, 12	keins	Parallelbetrieb Telefonieren nicht bei CSD; bei GPRS wird gleiche Leitung benutzt

Sensitive Entscheidungen: 7 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
7	Parallelbetrieb Telefonieren nicht bei CSD; bei GPRS wird gleiche Leitung benutzt	Telefonieren nicht nötig, wenn Daten übertragen werden → Response wird nicht unterstützt	2	`Communication` –Komponente müsste geändert werden	1	[+] Modifizierbarkeit	ja

Zusammenfassung
Usability-Problem: Telefonieren nicht möglich, wenn Daten übertragen werden. Response wird nicht unterstützt → 2

SA-Sensitivität: Interne Änderung der `Communication`-Komponente → Änderungsaufwand ist 1.

Tradeoffs: [+] Modifizierbarkeit durch schon vorhandene Kapselung
Andere(s) Szenario(s): k.A.
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,1)
Ziele: ASL 1. Iteration: (2,1); ASL 2. Iteration: (2,1) oder (0,0) – siehe Kommentar.

Fragen an Usability-Evaluation: Während der Benutzung des Programms die Nummer des Mobiltelefons wählen.
Sonstige Kommentare: Abrechnungsmodelle für GPRS oder 3G (z.B. UMTS) notwendig, mit denen ebenfalls eine komplette Kostenübernahme möglich ist – nur dann Änderung möglich und sinnvoll.

Abbildung 57: Analyseformular Using Applications Concurrently

5.3.5.7 *Maintaining Device Independence*

Maintaining Device Independence

Allgemeine Informationen Nummer: 26 Kurzname: Non-conflicting Device Usage Anforderungskategorie: Robustheit Interaktionskategorie: Exchange/Cooperation Within and Between Systems	**In die Analyse aufgenommen via (zutreffendes unterstreichen)** Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Rating mit Begründung** Architekt: vielleicht (GPS-Geräte ergänzen) Analyst 1: ja Analyst 2: ja Katalog: ●●●● High (based on literature)
Szenario Umwelt: Automobil Initiator: „Users“ Stimulus: „install/use an additional device “ Umgebung: Mobiles Endgerät Response: „The System keeps severity and frequency of device and software conflicts low.“ Respones Measure: „Within a given time span, a certain number of occurring conflicts with the additional device is not exceeded. (A) A certain number of conflicts with the device is not exceeded in total. (A) The problems caused by occurring conflicts do not prevent users from using the device with the application. (A)	**Rationale, Referenzen** Usability-Attribute: Effectiveness, Safety Usability-Anforderung: HCI-Rational: HCI-Patterns: [BJK01] 6 – *Maintaining Device Independence*: A user attempts to install a new device. Systems should be designed to reduce the severity and frequency of device conflicts. When device conflicts occur, the system should provide the information necessary to either solve the problem or seek assistance. SE-Rational: SE-Patterns: Projektreferenzen:

Analyse
Use Case(s): Anschluss von unterstützten (GPS-Empfänger) oder nicht unterstützten Geräten an das Handy

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
8	Maintaining Device Independence	Integration mit externen Programmen	k.A.	k.A.	keins	Integration mit externem GPS-Empfänger

Sensitive Entscheidungen: 8 (gut)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
8	Integration mit externem GPS-Empfänger	darf ordnungsgemäßen Betrieb nicht beeinträchtigen → Response realisiert	0	keine	0	[+] Modifizierbarkeit	nein

Zusammenfassung
Usability-Problem: Anschluss eines externen GPS-Empfängers ist möglich → 0

SA-Sensitivität: keine → 0

Tradeoffs: [+] Modifizierbarkeit, durch die leichte Ergänzung externer Geräte
Andere(s) Szenario(s): Feedback
Architekturdiagramm(e): k.A.

Fazit: Szenario wird unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (0,0)
Ziele: ASL 1. Iteration: (0,0), ASL 2. Iteration: (0,0)

Usability-Evaluation: Test, ob und wie einfach externe GPS-Geräte angeschlossen werden können.
Sonstige Kommentare: keine.

Abbildung 58: Analyseformular Maintaining Device Independence

5.3.5.8 *Progress Indication*

Progress Indication

Allgemeine Informationen
Nummer: 35
Kurzname: Progress Indication
Anforderungskategorie: Intuition
Interaktionskategorie(n): Orientation; Executing, Repeating and Revoking Commands

In die Analyse aufgenommen via (zutreffendes unterstreichen)
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Rating mit Begründung
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „Users“
Stimulus: „want to be informed about an operation's progress and/or its remaining duration“
Umgebung: Mobiles Endgerät
Response: „The operation's progress, along with time information, is constantly communicated by an adequate interface element.“
Response Measure: „The system calculates how long the operation is still going to take and present the result to the users. (A) Calculated time span and actual remaing duration differ by no more than a certain value. (A) Users can tell approximately by the number or the graphic provided how long the operation is still going to take. (U)“

Rationale, Referenzen
Usability-Attribute: Efficiency, Learnability
Usability-Anforderung:
HCI-Rational:
HCI-Patterns: [BJK01] 19 – *Predicting Task Duration:* A user may want to to make informed decisions about what to do while a system completes a long running operation. Systems should present expected task durations. [Ti05] 49 – *Progress Indicator:* Show the user how much progress was made on a time-consuming operation.
SE-Rational:
Projektreferenzen: keine.

Analyse
Use Case(s): Client verliert die Verbindung

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
9	Progress Indication	Anzeige des Sendens, Anzeige des Suchlaufs für externe GPS-Geräte	`TSEngine`, `TSAppUI`	1, 2, 3, 6, 7, 8, 9	keins	Keine Anzeige des Fortschritts

Sensitive Entscheidungen: 9 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
9	Keine Anzeige des Fortschritts	Information über Status des Systems notwendig → Response nicht unterstützt	2	Status muss angezeigt werden	1	k.A.	ja

Zusammenfassung
Usability-Problem: Information über den Status des Systems ist notwendig, Response wird nicht unterstützt → 2

SA-Sensitivität: Verhalten bestehender Komponenten muss geändert werden, im Detail ist der Änderungsaufwand nicht bekannt, aber mit Sicherheit nicht sehr komplex, da TSEngine alle Daten vorliegen haben müsste und von uns „nur“ interne Änderungen erwartet werden. → 1

Tradeoffs: k.A.
Andere(s) Szenario(s): Feedback
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,1)
Ziele: ASL 1. Iteration: (0,0) – hohe Priorität, ASL 2. Iteration: (0,0)

Usability-Evaluation: Nach der nächsten Iteration – erkennen Benutzer, wie lang eine Interaktion dauert?
Sonstige Kommentare: keine

Abbildung 59: Analyseformular Progress Indication

5.3.5.9 *Familiar Appearance and/or Behavior*

Familiar Appearance and/or Behavior

Allgemeine Informationen Nummer: 37 Kurzname: Familiar Appearance and/or Behaviour Anforderungskategorie: Intuition Interaktionskategorie: Using an Unfamiliar System, Adaptation (by Users/to Users/for Tasks)	**In die Analyse aufgenommen via** Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes **Begründung der Auswahl** Architekt: ja Analyst 1: ja (Softkeys) Analyst 2: ja Katalog: ●●●● High (based on literature)
Szenario Umwelt: Automobil Initiator: „Users“ Stimulus: „want to use (part of) an application they are not familiar with.“ Umgebung: Mobiles Endgerät Response: „Appearance and/or behaviour of the new software are designed consistently with the software the users are familiar with.“ Response Measure: „Users use the new or changed software successfully without learning a new interface concept. (U) The system behaves as the users expect it to. (U)“	**Rationale, Referenzen** Usability-Attribute: Effectiveness, Efficiency, Learnability, Memorability HCI-Rational: HCI-Patterns: [BJK01] 13 – *Leveraging Human Knowledge*: A user is required to use a new application on a familiar platform, a new version of a familiar application or a new product in an established product line. System designers should strive to develop upgrades that leverage users' knowledge of prior systems and allow them to move quickly and efficiently to the new system. [Fo05] 30 – *Emulation:* Emulate the appearance and/or behavior of a different system. SE-Rational: SE-Patterns: Projektreferenzen: bisher keine.

Analyse
Use Case(s): Stoppen einer Navigation (bei beiden Anwendungen)

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
10	Familiar Appearance and/or Behavior	Navigieren (Hauptinteraktion)	`TSAppUI, Platform UI Umgebung`	1	keins	Unterstützung der plattformtypischen Softkeys, Anwendung der entsprechenden Styleguides

Sensitive Entscheidungen: 10 (gut)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
10	Unterstützung der plattformtypischen Softkeys, Anwendung der entsprechenden Styleguides	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein

Zusammenfassung
Usability-Problem: Response wird unterstützt. → 0

SA-Sensitivität: keine → 0.

Tradeoffs: k.A.
Andere(s) Szenario(s): k.A.
Architekturdiagramm(e): k.A.

Fazit: Szenario wird unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (0,0)
Ziele: ASL 1. Iteration: (0,0) ; ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: keine.
Sonstige Kommentare: keine.

Abbildung 60: Analyseformular Familiar Appearance and/or Behavior

5.3.5.10 *Verifying Resources*

Verifying Ressources

Allgemeine Informationen
Nummer: 41
Kurzname: Verifying Ressources
Anforderungskategorie: Robustheit
Interaktionskategorie: Executing, Repeating and Revoking Commands

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „Users“
Stimulus: „want to start an operation that needs a set/amount of resources that may not be fully available. “
Umgebung: Mobiles Endgerät
Response: „The system only starts the operation if all of the required resources are available at that time.“
Response Measure: n/a

Rationale, Referenzen
Usability-Attribute: Efficiency, Safety
HCI-Rational:
HCI-Patterns: [BJK01] 23 – *Verifying Resources:* An application may fail to verify that necessary resources exist before beginning an operation. Applications should verify that all necessary resources are available before beginning an operation.
SE-Rational:
SE-Patterns:
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Starten der Anwendung

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
11	Verifying Ressources	Starten der Anwendung	`TSEngine`	zum System	keins	Start der Anwendung nur, wenn genug Ressourcen vorhanden sind

Sensitive Entscheidungen: 11 (gut)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
11	Start der Anwendung nur, wenn genug Ressourcen vorhanden sind	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein

Zusammenfassung
Usability-Problem: Keins, die Response wird unterstützt. → 0

SA-Sensitivität: keine Änderungen nötig → 0.

Tradeoffs: k.A.
Andere(s) Szenario(s): k.A.
Architekturdiagramm(e): k.A.

Fazit: Szenario wird unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (0,0)
Ziele: ASL 1. Iteration: (0,0); ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Szenarios mit Programmstart beginnen.
Sonstige Kommentare: keine.

Abbildung 61: Analyseformular Verifying Resources

5.3.5.11 *Error Messages*

Error Messages

Allgemeine Informationen
Nummer: 56
Kurzname: Error Messages
Anforderungskategorie: Robustheit
Interaktionskategorie: Structuring and Displaying Content/Information/Data, Error Handling and Help

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●○ High

Szenario
Umwelt: Automobil
Initiator: „System“
Stimulus: „throws an error message“
Umgebung: Mobiles Endgerät
Response: „An appropriate style is used to present messages of this kind on a mobile device. Users are given the necessary information to cope with the problem.“
Response Measure: „Users comprehend the information given. (U)“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Safety, Learnability
HCI-Rational:
HCI-Patterns: [Li09] 10 – Error Messages
SE-Rational:
SE-Patterns:
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Rückmeldung bei Fehlern, z.B. Verlust der GPS-Verbindung, Nachricht wurde nicht gesendet

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
12	Error Messages	Rückmeldung bei Fehlern	`TSAppUI, TSEngine,`	2, 3	Blackboard	Existenz von Fehlerdefinitionen und Exception Handling
13	Error Messages	Rückmeldung bei Fehlern	`TSAppUI, TSEngine,`	2, 3	Blackboard	Existenz von Fehlermeldungen mit Informationen, was getan werden muss
14	Error Messages	Rückmeldung bei Fehlern	`TSEngine, SatPos.Sys.`	6, 7	Blackboard	Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode
15	Error Messages	Rückmeldung bei Fehlern	`TSEngine, Communic.`	8, 9	Blackboard	Communication-Komponente meldet Verfügbarkeiten der Verbindungen

Sensitive Entscheidungen: 12 (gut), 13 (gut), 14 (gut), 15 (gut)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
12	Existenz von Fehlerdefinitionen und Exception Handling	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
13	Existenz von Fehlermeldungen mit Informationen, was getan werden muss	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
14	Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
15	Communication-Komponente meldet Verfügbarkeiten der Verbindungen	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein

Zusammenfassung

Abbildung 62: Analyseformular Error Messages, Seite 1

SA-Sensitivität: Keine → Änderungsaufwand 0.

Tradeoffs: [+] Fehlertoleranz: auftretende Fehler bei der Benutzung und des Systems werden erkannt und behandelt
Andere(s) Szenario(s): k.A.
Architekturdiagramm(e): k.A.

Fazit: Szenario wird unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (0,0)
Ziele: ASL 1. Iteration: (0,0), ASL 2. Iteration: (0,0)

Fragen an Usability-Evaluation: Welche Fehler passieren bei der Benutzung? Wurden alle definiert und werden alle abgefangen?
Sonstige Kommentare: keine.

Abbildung 63: Analyseformular Error Messages, Seite 2

5.3.5.12 *Help*

Help

Allgemeine Informationen
Nummer: 76
Kurzname: Multi-Level Help
Anforderungskategorie: Robustheit
Interaktionskategorie: Error Handling and Help

In die Analyse aufgenommen via
Kontextfaktoren, Walkthrough, Brainstorming, Generisches Szenario, Eigenes

Begründung der Auswahl
Architekt: ja
Analyst 1: ja
Analyst 2: ja
Katalog: ●●●● High (based on literature)

Szenario
Umwelt: Automobil
Initiator: „Users“
Stimulus: „need help with the system“
Umgebung: Mobiles Endgerät
Response: „The system immediately provides the appropriate kind of assistance.“
Response Measure: „Users comprehend the help given (U) and act appropriately (U).“

Rationale, Referenzen
Usability-Attribute: Effectiveness, Efficiency, Safety
HCI-Rational:
HCI-Patterns: [Ti05] 20 – *Multi-Level Help:* Use a mixture of lightweight and heavyweight help techniques to support users with varying needs., [BJK01] 10 – *Providing Good Help:* A user needs help. Help procedures should be context dependent and sufficiently complete to assist users in solving problems. [Fo05] 31 – *Context Sensitive Help:* Monitor what the user is currently doing, and make documentation available that is relevant to the completion of the task.
SE-Rational:
SE-Patterns:
Projektreferenzen: bisher keine.

Analyse
Use Case(s): Nutzer ruft Hilfe vor Fahrtbeginn auf

Nr.	Sz.	Use Case	Komponenten	Schnittstellen	Pattern	Response betreffende Entscheidung (Sensitivitätspunkt)
16	Help	Navigieren (Hauptinteraktion)	nicht existente Hilfskomponente und `TS-Engine`	keine vorhanden	keins	Nichtexistenz einer spezialisierten Hilfskomponente

Sensitive Entscheidungen: 16 (kritisch)

S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
16	Nichtexistenz einer spezialisierten Hilfskomponente	es gibt keine Hilfe, aber online als PDF Installations- und Bedienungsanleitungen → Response nicht realisiert	2	Hilfskomponente muss ergänzt werden	2	k.A.	ja

Zusammenfassung
Usability-Problem: Es gibt keine Hilfe oder Anleitung, so dass die Benutzer bei Fragen zur Benutzung und Fehlern beim Service anrufen müssen oder online Hilfe suchen müssen → 2

SA-Sensitivität: Eine Komponente muss ergänzt und integriert werden, es muss entschieden werden, wann Hilfe angeboten wird und welche Texte formuliert werden müssen, es gibt außerdem auch keine Screens → Änderungsaufwand ist 1 oder 2.

Tradeoffs: k.A.
Andere(s) Szenario(s): k.A.
Architekturdiagramm(e): k.A.

Fazit: Szenario wird nicht unterstützt.
ASL 0: Aktueller Einfluss des Szenarios auf (Usability, Softwarearchitektur): (2,2)
Ziele: ASL 1. Iteration: (2,2); ASL 2. Iteration: (0,0) (die Anforderung ist nicht wichtig)

Fragen an Usability-Evaluation: Fällt den Benutzern überhaupt die fehlende Hilfe auf?
Sonstige Kommentare: Anhand der FAQ und anhand von Tests kann die notwendige Unterstützung der Benutzer durch eine kontextsensitive

Abbildung 64: Analyseformular Help

5.3.6 *Durchführung der Fallstudie TS: SATURN-Phase 4*

5.3.6.1 *Architektur-Support-Level*

Szenarios	Begründung	Sensitive Entscheidungen	Unterstützung des Szenarios	Max. Einfluss auf Usability, Softwarearchitektur (ASL aktuell)	ASL0 (erste Analyse)	ASL1 (folgende Iteration)	ASL 2 (folgende Iteration)
Observing System State (System Feedback)	Walkthrough	S1 (k)	nein	2,2	2,2	0,0	0,0
Abbrechen (Canceling Commands)	Kontext: Walkthrough	S2 (k)	nein	2,2	2,2	0,0	0,0
Forgiving Format	Walkthrough	S3 (k)	nein	2,1	2,1	2,1	0,0
Checking for Correctness	Walkthrough	S4 (k)	nein	1,1	1,1	1,1	0,0
Context-based Workflow	Hauptszenario	S5, S6 (k)	teilweise (GPS-Ausfall probl.)	2,1	2,1	0,0	0,0
Using Applications Concurrently	Katalog	S7 (k)	nein	2,1	2,2	2,2	2/2 oder 0,0*
Maintaining Device Independence	Katalog	S8	ja	0,0	0,0	0,0	0,0
Progress Indication	Katalog	S9 (k)	nein	2,1	2,1	0,0	0,0
Familiar Appearance and/or Behaviour	Katalog	S10	ja	0,0	0,0	0,0	0,0
Verifying Resources	Katalog	S11	ja	0,0	0,0	0,0	0,0
Error Messages	Katalog	S12, S13, S14, S15	ja	0,0	0,0	0,0	0,0
Help	Katalog	S16	nein	2,2	2,2	2,2	0,0

Einfluss auf Usability
0... Die Anforderung wird umgesetzt.
1... Die Anforderung wird teilweise umgesetzt.
2... Die Anforderung wird nicht umgesetzt.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.

Einfluss auf Softwarearchitektur
0... Keine Modifikationen werden erwartet.
1... Einfache Modifikationen werden erwartet.
2... Komplexe Modifikationen werden erwartet.
3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

* abh. von Abrechnungsmodellen der Provider

Abbildung 65: ASL TrafficScanner

Zwölf Interaktionsszenarios wurden ausgewählt, um die SA zu analysieren. Vier davon wurden aufgrund des Walkthroughs bestimmt (System Feedback, Abbrechen, Forgiving Formats, Checking for Correctness), eines beschreibt die Hauptinteraktion (Context-based Workflow), sieben Interaktionsszenarios wurden aus dem Interaktionsszenariokatalog entnommen.

Vier Interaktionsszenarios werden komplett unterstützt, eins kann teilweise, sieben können nicht durchgeführt werden. Davon werden für die drei Interaktionsszenarios System Feedback, Abbrechen und Hilfe strukturelle Änderungen vorgeschlagen; für die vier Interaktionsszenarios Forgiving Format, Checking for Correctness, Context-based Workflow und Progress Indication reichen voraussichtlich einfache Modifikationen. Die vier Interaktionsszenarios Maintaining Device Independence, Familiar Appearance and/or Behaviour, Verifying Resources und Error Messages werden unterstützt und erfordern deshalb keine Änderungen. Insgesamt ist dies ein gutes Ergebnis für die Softwarearchitektur dieser mobilen Anwendung.

5.3.6.2 *Architekturentscheidungen auflisten*

Die Teilnehmer ermittelten 16 szenario-beeinflussende Entscheidungen. Acht realisieren die Response nicht oder nur teilweise. Die Abbildungen 66, 67, 68 listen alle szenario-beeinflussenden Entscheidungen und ihre Beurteilungen auf.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Intention	Feedback	ASL 1	1	Nichtexistenz eines Status "ich versende gerade" für Feedback oder eine „Meldungswarteschlange“	User sehen nicht, ob Meldung noch in Warteschlange ist. Sie sollten es aber sehen., Feedback über Warteschlange → Response nicht unterstützt	2	TSAppUI, TSEngine, Comm. und Interface sollten angepasst werden;	2	[-] Fehlertoleranz	ja
Robustheit	Canceling Commands	ASL1	2	Cancel des Sendens wird architekturell nicht unterstützt	Abbruch des Sendens wird architekturell nicht unterstützt → Response nicht unterstützt	2	Änderungen in mehreren Komponenten nötig, evtl. Cancel-Komponente, Koop. mit Plattform ist möglich (CActive-Framework) wird aber aktuell nicht unterstützt	2	k.A.	ja
Robustheit	Forgiving Format	ASL2	3	Feedback bei Fehler: Eingabe der Staumelder-nummer unter 6 Stellen mit führender Null	Eingabefehler wird nicht automatisch korrigiert, oder angezeigt → Response wird nicht unterstützt	2	Änderung der Behandlung der Eingabedaten in den bestehenden Komponenten. Ergänzen von Dialog und entsprechenden Funktionen.	1	keine	ja

Abbildung 66: szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 1/3

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Checking for Correctness	ASL2	4	Speicherung der Login-Daten (eine Zahl) als Integer	Eingabe wird auf 6-stellige Nummer geprüft, falls Login 5-stellig muss laut FAQ vorangehende Null ergänzt werden. Response wird umgesetzt, aber falsch. Integer-Werte speichern vorangehende Nullen nicht; Benutzerfehler und Mehrfacheingabe wird provoziert → Response wird teilweise unterstützt	1	Änderung in TSAppUI	1	keine	ja
Intuition	Context-based Workflow	ASL 1	5	System fordert Benutzereingabe basierend auf aktueller Geschwindigkeit	Kein negativer Einfluss. Response wird unterstützt.	0	Keine.	0	keine	nein
			6	Ablauflogik bei Staumeldung: aktuelle Daten werden verwendet, statt die, die zur Stauerkennung geführt haben	Fehler provoziert, falls keine GPS-Daten in dem Moment vorliegen Stauerkennung-User soll bestätigen - dann kann es doch nicht gesendet werden: doppelte Arbeit nötig, Erwartungen nicht erfüllt. Staumeldung auch doppelt falls kein Empfang. Workflow wird also im Fehlerfall nicht beendet → Response nicht unterstützt	2	TSEngine, JamDetector müßten Daten speichern, die zur Stauerkennung geführt haben.	1	[-] Fehlertoleranz des Systems	ja
Robustheit	Using Applications Concurrently	ASL 3	7	Parallelbetrieb Telefonieren nicht bei CSD; bei GPRS wird gleiche Leitung benutzt	Telefonieren nicht nötig, wenn Daten übertragen werden → Response wird nicht unterstützt	2	Communication – Komponente müsste geändert werden	1	[+] Modifizier-barkeit	ja
Robustheit	Maintaining Device Inependence	ASL0	8	Integration mit externem GPS-Empfänger	darf ordnungsgemäßen Betrieb nicht beeinträchtigen → Response realisiert	0	keine	0	[+] Modifizier-barkeit	nein
Intuition	Progress Indication	ASL1	9	Keine Anzeige des Fortschritts	Information über Status des Systems notwendig → Response nicht unterstützt	2	Status muss angezeigt werden	1	k.A.	ja
Intuition	Familiar Appearance and/or Behavior	ASL0	10	Unterstützung der plattformtypischen Softkeys, Anwendung der entsprechenden Styleguides	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein
Robustheit	Verifying Resources	ASL0	11	Start der Anwendung nur, wenn genug Ressourcen vorhanden sind	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein
Robustheit	Error Messages	ASL0	12	Existenz von Fehlerdefi-nitionen und Exception Handling	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertole-ranz	nein
			13	Existenz von Fehlermel-dungen mit Informatio-nen, was getan werden muss	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
			14	Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
			15	Communication-Komponente meldet Verfügbarkeiten der Verbindungen	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein

Abbildung 67: szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 2/3

Robustheit	Help	ASL2	16	Nichtexistenz einer spezialisierten Hilfskomponente	es gibt keine Hilfe, aber online als PDF Installations- und Bedienungsanleitungen → Response nicht realisiert	2	Hilfskomponente muss ergänzt werden	2	k.A.	ja

Einfluss auf Usability	Einfluss auf Softwarearchitektur
0... Die Anforderung wird umgesetzt.	0... Keine Modifikationen werden erwartet.
1... Die Anforderung wird teilweise umgesetzt.	1... Einfache Modifikationen werden erwartet.
2... Die Anforderung wird nicht umgesetzt.	2... Komplexe Modifikationen werden erwartet.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.	3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

Abbildung 68: szenario-beeinflussende Entscheidungen und ihre Bewertung, Seite 3/3

5.3.6.3 *Themen beschreiben*

Die Architekturentscheidungen werden in folgende thematische Gruppen agglomeriert und danach abgehandelt.

Architekturentscheidung(en)	*Thema*
S1, S5, S6, S9, S14	Staumeldungen (Hauptinteraktion)
S2	Abbrechen
S3, S4	Login-Probleme (Nutzeranfragen)
S7	Gleichzeitige Nutzung von Programmen
S8	Externe GPS-Geräte ergänzen
S10	Plattformspezifische Styleguides
S11	Gesicherter Start der Anwendung
S12, S13, S14, S15	Fehlermeldungen
S16	Kontext-sensitive Hilfe

Tabelle 35: Clustering von Architekturentscheidungen in Themen

STAUMELDUNGEN (HAUPTINTERAKTION) Alle Architekturentscheidungen zum Thema Staumeldungen sind in Tabelle 69 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Error Messages	ASL0	14	Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
Intention	Feedback	ASL 1	1	Nichtexistenz eines Status "ich versende gerade" für Feedback oder eine „Meldungswarteschlange“	User sehen nicht, ob Meldung noch in Warteschlange ist. Sie sollten es aber sehen., Feedback über Warteschlange → Response nicht unterstützt	2	TSAppUI, TSEngine, Comm. und Interface sollten angepasst werden;	2	[-] Fehlertoleranz	ja
Intuition	Context-based Workflow	ASL 1	5	System fordert Benutzereingabe basierend auf aktueller Geschwindigkeit	Kein negativer Einfluss. Response wird unterstützt.	0	Keine.	0	keine	nein
			6	Ablauflogik bei Staumeldung: aktuelle Daten werden verwendet, statt die, die zur Stauerkennung geführt haben	Fehler provoziert, falls keine GPS-Daten in dem Moment vorliegen Stauerkennung-User soll bestätigen - dann kann es doch nicht gesendet werden: doppelte Arbeit nötig, Erwartungen nicht erfüllt. Staumeldung auch doppelt falls kein Empfang. Workflow wird also im Fehlerfall nicht beendet → Response nicht unterstützt	2	TSEngine, JamDetector müßten Daten speichern, die zur Stauerkennung geführt haben.	1	[-] Fehlertoleranz des Systems	ja
Intuition	Progress Indication	ASL1	9	Keine Anzeige des Fortschritts	Information über Status des Systems notwendig → Response nicht unterstützt	2	Status muss angezeigt werden	1	k.A.	ja

Abbildung 69: Entscheidungen S1, S5, S6, S9 und S14 betreffen die Staumeldungen

Erläuterung: Das User Interface fragt den Nutzer, ob er einen Stau melden möchte, sobald das Fahrzeug eine festgelegte Geschwindigkeit unterschreitet (S5). Die Anwendung sammelt weiterhin jede Sekunde GPS-Koordinaten (S14) und verwendet die aktuellsten Daten für die Staumeldung (S6).

Allerdings kann im User Interface nicht angezeigt werden, dass eine Meldung gerade gesendet wird, da es keinen Status für das Versenden einer Nachricht oder eine Queue für Meldungen gibt (S1). Es gibt auch keine Anzeige des Sendefortschritts (S9). Falls der Nutzer zudem etwas Zeit braucht, um auf den Button zu klicken, und das Satellitenpositionssystem dann einen Fehlercode liefert, kann keine Meldung versendet werden (ebenso S6).

Voraussichtliche Unterstützung für die Usability:

(+) Automatische Ermittlung der Geschwindigkeit (Kontext) durch GPS-Komponente und von dieser Kontextänderung abhängiges Triggern einer Interaktion, führen zu einer sehr schnellen und unkomplizierten Interaktion: Das System meldet sich beim Nutzer, sobald er langsamer wird (S5, S6, S14). Der Nutzer kann sich also auf das Fahren konzentrieren und muss nicht extra eine Staumeldung starten; die Staumeldung selbst passiert im günstigsten Fall mit einem Tap auf einen Button. Danach lässt die Anwendung den Nutzer wieder „in Ruhe“. Dies ist einfach, effizient und effektiv.

(+) Separation von Komponenten zur Verarbeitung der Daten und der Erhebung der Daten ermöglicht, dass potenziell auch andere Möglichkeiten zur Bestimmung der Geschwindigkeit verwendet werden können: Die Verantwortlichkeiten der Komponenten `TrafficScanner Engine`, `JamDetector` und `SatPositionSystem` sind so geregelt, dass es möglich ist, dass andere Systeme zur Positionsbestimmung angebunden werden können (S14), siehe auch Abbildung 45, S. 127.

Hinweise auf mögliche Probleme bei der Benutzung:

(-) GPS-Daten, die zu einer Staumeldung geführt haben oder die Meldungen selbst werden nicht gespeichert. Dies verhindert eine stabile, von Funklöchern unabhängige Staumeldung: Falls die Netzverbindung eine Sekunde lang nicht funktioniert, kann keine Meldung abgeschickt werden (S6). Das System lenkt also den Nutzer vom Fahren ab, er interagiert mit der Anwendung, nur um eine Fehlermeldung angezeigt zu bekommen, dass die Meldung doch unmöglich ist. Ein technischer Fehler ist offensichtlich, das System kann seine Hauptaufgabe nicht erfüllen und hat den Anwender außerdem unnötig beim Steuern seines Fahrzeuges unterbrochen.

(-) Weder eine Meldungs-Queue zum Zwischenspeichern von Meldungen noch ein Status „Sende gerade“ wurden integriert, so dass Nutzer kein Feedback zum Sendestatus erhalten: Das User Interface sollte anzeigen, was das System gerade macht, insbesondere, dass eine Meldung gerade gesendet wird. Nur durch entsprechendes Vorhalten und Weiterleiten dieser Informationen – nach S1 und S9 geschieht das nicht – wäre das möglich.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(+) Modifizierbarkeit: Durch die Separation der Komponenten können grundsätzlich andere Methoden zur Ortsbestimmung integriert werden.

(-) Fehlertoleranz: Die Abhängigkeit von den fünf aktuellen und nicht zwischengespeicherten Positionsdaten ist sehr kritisch, da die Anwendung schon bei einem kurzen Ausfall eines angeschlossenen Systems oder bei schlechter Netzabdeckung nicht mehr funktioniert.

Mögliche Modifikationen:

- Andere oder zusätzliche Verfahren zur Ortsbestimmung können kritische Ausfälle verhindern: Es gibt andere Möglichkeiten der Ortsbestimmung die zusätzlich oder alternativ in die Anwendung integriert werden können. Dies erfordert voraussichtlich Änderungen an einer Komponente (`TSEngine` und eine oder mehr neue Komponente(n) für die Ortsbestimmung(en) sowie Schnittstellen (Änderungskomplexität 2).

- Die Schnittstelle `ComChannelObserver` sollte um den Status „sende gerade“ erweitert werden, damit das User Interface diesen Status anzeigen kann: Die Nutzer sollen Feedback darüber erhalten, was das System gerade macht (Interaktionsszenario System Feedback). Der Status soll über die Komponenten `TSEngine` und `TSAppUI` im User Interface angezeigt werden. Dies erfordert also Änderungen innerhalb der genannten architektonischen Elemente (Änderungskomplexität 1).

- Die Komponenten `TSEngine` und `TSJamDetector` sollen es ermöglichen, die Daten für eine Staumeldung zumindest kurzzeitig zu speichern, damit die Staumeldung mit diesen Daten gesendet werden kann: werden die Daten vorgehalten, die dazu geführt haben, dass sich die Anwendung beim Nutzer meldet, wird abgesichert, dass genug Werte für die Staumeldung vorliegen. Dass es sich um Ortsdaten handelt, die eine oder mehrere Sekunden alt sind, sollte bei einem Stau nebensächlich und vertretbar sein, da sich das Fahrzeug dann noch nicht sehr weit bewegt haben kann. Dies erfordert Änderungen innerhalb der zwei genannten Komponenten (Änderungskomplexität 1).

ABBRECHEN Alle Architekturentscheidungen zum Thema Abbrechen sind in Tabelle 70 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Canceling Commands	ASL1	2	Cancel des Sendens wird architekturell nicht unterstützt	Abbruch des Sendens wird architekturell nicht unterstützt → Response nicht unterstützt	2	Änderungen in mehreren Komponenten nötig, evtl. Cancel-Komponente, Koop. mit Plattform ist möglich (CActive-Framework) wird aber aktuell nicht unterstützt	2	k.A.	ja

Abbildung 70: S2 betrifft das Thema Abbrechen

Erläuterung: Die vorhandenen Daten enthielten keinerlei Hinweis darauf, dass das Abbrechen implementiert wurde (S2).

Hinweise auf mögliche Probleme bei der Benutzung:

(-) Ohne Implementierung von Abbrechen ist es unmöglich, die Anmeldung oder das Senden abzubrechen: Das Abbrechen ist nicht im Rahmen der App möglich (S2), es wäre also nötig, die App zu schließen und neu zu starten, um das Senden oder das Login abzubrechen. Wünschenswert ist ein Abbruch aus Nutzersicht, da er grundsätzlich die Kontrolle über das Verhalten des mobilen Endgerätes haben will. Sollte er aus irgendeinem Grund eine Interaktion abbrechen wollen, so sollte es möglich sein (siehe Interaktionsszenario Abbrechen).

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(-) Fehlertoleranz: Da es nicht möglich ist, Prozesse abzubrechen (S2), muss die Anwendung geschlossen werden, um fehlerhafte oder unerwünschte Prozesse zu beenden.

Mögliche Modifikationen:

- Framework zum Abbrechen implementieren, um das Interaktionsszenario Abbrechen durchführen zu können: Die mobile Plattform bietet ein Framework zum Abbruch von Befehlen an, allerdings muss die Funktionalität in der Software durch die Entwickler implementiert werden. Es ist ein Querschnittsthema, wobei Änderungen voraussichtlich an verschiedenen Komponenten notwendig sind. Es ist auch möglich, dass eine neue Komponente integriert werden muss, die das Abbrechen regelt. Diese Änderung betrifft mehrere Komponenten und ist aufwendig (Änderungskomplexität 2).

LOGIN-PROBLEME (NUTZERANFRAGEN) Alle Architekturentscheidungen zum Thema Login-Probleme sind in Tabelle 71 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Forgiving Format	ASL2	3	Feedback bei Fehler: Eingabe der Staumeldernummer unter 6 Stellen mit führender Null	Eingabefehler wird nicht automatisch korrigiert, oder angezeigt → Response wird nicht unterstützt	2	Änderung der Behandlung der Eingabedaten in den bestehenden Komponenten. Ergänzen von Dialog und entsprechenden Funktionen.	1	keine	ja
Robustheit	Checking for Correctness	ASL2	4	Speicherung der Login-Daten (eine Zahl) als Integer	Eingabe wird auf 6-stellige Nummer geprüft, falls Login 5-stellig muss laut FAQ vorangehende Null ergänzt werden. Response wird umgesetzt, aber falsch. Integer-Werte speichern vorangehende Nullen nicht; Benutzerfehler und Mehrfacheingabe wird provoziert → Response wird teilweise unterstützt	1	Änderung in TSAppUI	1	keine	ja

Abbildung 71: S3 und S4 betreffen das Thema Login-Probleme

Erläuterung: Es wird geprüft, ob die eingetragene Benutzer-Identifikationsnummer (ID) genau sechs Stellen hat (S3). Die Analyse der Architektur zeigte, dass diese Nummer als Integer abgespeichert wird (S4), dass vorangestellte Nullen also nicht gespeichert werden und damit für die Anwendung irrelevant sind.

Hinweise auf mögliche Probleme bei der Benutzung:

(-) Laut Analysekontext haben manche Nutzern-IDs weniger als sechs Stellen. Wenn sie sich einloggen wollen, wird eine eine Fehlermeldung angezeigt: „Bitte eine sechsstellige Nummer angeben!“. Dieser vom System provozierte Fehler führte zu häufigen Nachfragen beim Service, der eine vorangehende Null empfiehlt. Nur dann ist ein Login und nachfolgende Nutzung möglich.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(-) Fehlertoleranz: Nutzer können sich nicht mit korrekt eingegebenen IDs anmelden.

Mögliche Modifikationen:

- ID nicht auf Sechstelligkeit prüfen und falls notwendig beim Senden an den Server eine vorangehende Null ergänzen führt dazu, dass sich Nutzer mit kürzeren IDs anmelden können: Da in der Anwendung die ID als Integer abgespeichert wird (S4), werden vorangestellte Nullen entfernt. Deshalb ist es nicht nötig, die ID auf Sechstelligkeit zu prüfen. Die Komponenten `TSAppUI` und `TSEngine` sollten dahingehend überarbeitet werden, damit die Nutzer nicht mehr unnötig unterbrochen werden. Falls der Server eine sechsstellige ID fordert, kann hier die ID mit vorangestellter Null gesendet werden. Dazu sollte die Komponente `TSEngine` die ID aufbereiten und sie nicht als Integer an die Komponenten zur Kommunikation weiterleiten. Dies erfordert Änderungen innerhalb der zwei genannten Komponenten (Änderungskomplexität 1).

GLEICHZEITIGE NUTZUNG VON PROGRAMMEN Alle Architekturentscheidungen zum Thema Gleichzeitige Nutzung von Programmen sind in Tabelle 72 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Using Applications Concurrently	ASL 3	7	Parallelbetrieb Telefonieren nicht bei CSD; bei GPRS wird gleiche Leitung benutzt	Telefonieren nicht nötig, wenn Daten übertragen werden → Response wird nicht unterstützt	2	Communication – Komponente müsste geändert werden	1	[+] Modifizier-barkeit	ja

Abbildung 72: S7 betrifft das Thema Gleichzeitige Nutzung von Programmen

Erläuterung: Sobald Daten übertragen werden, ist Telefonieren nicht möglich (S7). Hinweise auf mögliche Probleme bei der Benutzung:

(-) Kein gleichzeitiges Staumelden und Telefonieren: Sobald die Anwendung sendet, kann das Telefon nicht mehr verwendet werden, um zu telefonieren. Der Hauptverwendungszweck des Telefons wird behindert. Allerdings sind die Komponente `Communication` und die entsprechenden Schnittstellen darauf vorbereitet, diese Funktionalität umzusetzen, wenn das Mobiltelefon sie anbietet.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(+) Modifizierbarkeit: Die Separation der Komponenten ermöglicht ein leichtes Ergänzen der gewünschten Funktionalität.

Mögliche Modifikationen:

- Überarbeitung der Komponente `Communication`, um gleichzeitig Staumelden und Telefonieren zu können: Dies erfordert unbestimmte Änderungen innerhalb der Komponente (Änderungskomplexität 1).

EXTERNE GPS-GERÄTE ERGÄNZEN Alle Architekturentscheidungen zum Thema Externe GPS-Geräte ergänzen sind in Tabelle 73 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Maintaining Device Inependence	ASL0	8	Integration mit externem GPS-Empfänger	darf ordnungsgemäßen Betrieb nicht beeinträchtigen → Response realisiert	0	keine	0	[+] Modifizierbarkeit	nein

Abbildung 73: S8 betrifft das Thema Externe GPS-Geräte ergänzen

Erläuterung: Externe GPS-Geräte können ergänzt werden (S8).
Voraussichtliche Unterstützung für die Usability:

(+) Externe GPS-Geräte zu ergänzen ermöglicht, dass Nutzer Geräte ihrer Wahl verwenden können: Die Anwendung funktioniert unabhängig von angeschlossenen Geräten korrekt, wobei die Abhängigkeit von funktionierenden Positionssystemen schon beschrieben wurde (siehe Thema Staumeldungen). Falls ein GPS-Gerät nicht funktioniert, kann es ausgetauscht werden.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(+) Modifizierbarkeit: Austauschbarkeit von externen Komponenten.

PLATTFORMSPEZIFISCHE STYLEGUIDES Alle Architekturentscheidungen zum Thema Plattformspezifische Styleguides sind in Tabelle 74 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Intuition	Familiar Appearance and/or Behavior	ASL0	10	Unterstützung der plattformtypischen Softkeys, Anwendung der entsprechenden Styleguides	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein

Abbildung 74: S10 betrifft das Thema Plattformspezifische Styleguides

Erläuterung: Die Funktionalität von Softkeys und die Vorgaben der Styleguides werden beachtet (S10).
Voraussichtliche Unterstützung für die Usability:

(+) Umsetzung plattformspezifischer Styleguides führt zu bekanntem Aussehen und Verhalten der Anwendung: Das Look and Feel einer Plattform wird für die Darstellung des User Interfaces und die Interaktionen verwendet (S10). Die Nutzer finden sich schnell zurecht und müssen nichts umlernen (siehe Begründung für das Interaktionsszenario Familiar Appearance and/or Behavior).

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen: keine genannt.

GESICHERTER START DER ANWENDUNG Alle Architekturentscheidungen zum Thema Gesicherter Start der Anwendung sind in Tabelle 75 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Verifying Resources	ASL0	11	Start der Anwendung nur, wenn genug Ressourcen vorhanden sind	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet.	0	Keine.	0		nein

Abbildung 75: S11 betrifft das Thema Gesicherter Start der Anwendung

Erläuterung: Die Anwendung startet nur, wenn genügend Ressourcen verfügbar sind (S11).

Voraussichtliche Unterstützung für die Usability:

(+) Prüfung von Ressourcen vor Start der Anwendung ermöglicht, diese zu starten und zu nutzen

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen: keine genannt.

FEHLERMELDUNGEN Alle Architekturentscheidungen zum Thema Fehlermeldungen sind in Tabelle 76 aufgeführt.

Fehlerdefinitionen im Allgemeinen

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Error Messages	ASL0	12	Existenz von Fehlerdefinitionen und Exception Handling	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
			13	Existenz von Fehlermeldungen mit Informationen, was getan werden muss	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
			14	Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein
			15	Communication-Komponente meldet Verfügbarkeiten der Verbindungen	Kein negativer. Entscheidung wird von beiden Beteiligten befürwortet. → Response wird unterstützt	0	Keine.	0	[+] Fehlertoleranz	nein

Abbildung 76: S12, S13, S14, S15 betreffen das Thema Fehlermeldungen

Erläuterung: Das Blackboard-Pattern [BMR+96] wird angewendet. Zu den Netzverbindungen liefert die Komponente `Communication` Statusinformationen (S15) an die Komponente `TSEngine`; die Komponente `SatPositionSystem` liefert ihr ggf. Fehlercodes. In der Komponente `TSEngine` werden Fehler auf Fehlertypen abgebildet (S12, S14, S15) und ggf. im User Interface über die Komponente `TSAppUI` als Fehlermeldungen angezeigt (S13). Diese Fehlermeldungen enthalten auch Informationen darüber, was der Nutzer tun kann (S13).

Voraussichtliche Unterstützung für die Usability:

(+) Einsatz des Blackboard-Patterns ermöglicht schnelle Fehlerbehandlung und fördert die Sicherheit bei der Benutzung: Wenn verschiedene vorab definierte Fehler auftreten, werden sie zentral gemeldet und behandelt, ggf. werden sie dem Nutzer mitgeteilt (S12, S13, S14, S15). Der Nutzer kann lernen, warum welche Probleme auftreten und erhält Informationen, wie er sie beheben kann bzw. was er als nächstes tun kann.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen:

(+) Fehlertoleranz: Auftretende Fehler bei der Benutzung sowie Fehler des Systems werden erkannt und behandelt.

KONTEXT-SENSITIVE HILFE Alle Architekturentscheidungen zum Thema Kontext-sensitive Hilfe sind in Tabelle 77 aufgeführt.

Anforderung	Szenario	Prio	S	Beschreibung	Pot. U-Problem	Grad	SA-Sensitivität	Grad	Tradeoffs +/-	Kritisch
Robustheit	Help	ASL2	16	Nichtexistenz einer spezialisierten Hilfskomponente	es gibt keine Hilfe, aber online als PDF Installations- und Bedienungsanleitungen → Response nicht realisiert	2	Hilfskomponente muss ergänzt werden	2	k.A.	ja

Abbildung 77: S16 betrifft das Thema Kontext-sensitive Hilfe

Erläuterung: Bis auf Installations- und Bedienungsanleitungen, die online auf der Anwendungswebsite zum Download angeboten werden (siehe Analysekontext dieser Fallstudie), gibt es keine (kontext-basierte) Hilfe (S16).
Hinweise auf mögliche Probleme bei der Benutzung:

(-) Ohne Hilfe bei Problemen werden Nutzer nicht gut angeleitet: Probleme haben größere Auswirkungen.

Wechselwirkungen zwischen Usability und anderen Qualitätsmerkmalen: es wurden keine genannt.
Mögliche Modifikationen:

- Integration einer neuen Komponente für die Hilfe: Eine Hilfskomponente sollte ergänzt und mit anderen verbunden werden. Der Aufruf sollte (wie im Interaktionsszenario gefordert) immer abhängig vom jeweiligen Problem des Nutzers sein. Diese Änderung betrifft verschiedene Komponenten. Verschiedene Optionen müssen noch bewertet werden (Änderungskomplexität 3).

SONSTIGES

- Fremdsprachige Versionen sind nicht geplant; evtl. für den inländischen Gebrauch dennoch gewünscht, z. B. auf Türkisch, Englisch (Rückfrage an Auftraggeber)

Beim Walkthrough wurden folgende Probleme in der Benutzerschnittstelle aufgedeckt:

- es wird zu viel und zu kleiner Text auf dem kleinen Bildschirm angezeigt (Lesbarkeit ist eingeschränkt, insbesondere für die Benutzergruppe Rentner),
- es werden Fachbegriffe aus der Anwendungsentwicklung verwendet, die Nutzer nicht gut verstehen können,

- kursive Schriftarten wurden eingesetzt, die generell sehr schwierig auf Bildschirmen zu lesen sind.

Die Kommentare für die User-Evaluation von TS werden in Tabelle 78 dargestellt.

Szenario	Fragen an Usability-Evaluation	Priorisierung
Observing System State (Feedback)	Testen des Hauptszenarios	ASL1
Cancel	Mit nächster Version: Beobachten und bewerten, wie die Benutzer die Verbindung abbrechen.	ASL1
Forgiving Format	Beobachten und bewerten, inwiefern die Benutzer mit dem Fehler zurecht kommen.	ASL2
Checking for Correctness	siehe Forgiving Format	ASL2
Context-based Workflow	Testen des Hauptszenarios	ASL1
Using Applications Concurrently	Während der Benutzung des Programms die Nummer des Mobiltelefons wählen.	ASL2
Maintaining Device Independence	Test, ob und wie einfach externe GPS-Geräte angeschlossen werden können.	ASL0
Progress Indication	Nach der nächsten Iteration – erkennen Benutzer, wie lang eine Interaktion dauert?	ASL1
Familiar Appearance and/or Behaviour	keine	
Verifying Ressources	Szenarios mit Programmstart beginnen.	ASL0
Error Messages	Welche Fehler passieren bei der Benutzung? Wurden alle definiert und werden alle abgefangen?	ASL0
Help	Fällt den Benutzern überhaupt die fehlende Hilfe auf?	ASL2

Abbildung 78: Fragen an Usability-Evaluation TrafficScanner

5.3.6.4 *Priorisierung der Interaktionsszenarios basierend auf Nutzungskontext*

1. Das Hauptszenario muss umgesetzt werden und auch im Fehlerfall funktionieren. Deshalb müssen die Interaktionsszenarios Context-based Workflow und Observing System State (Feedback), Progress Indicator und Cancel zuerst realisiert werden. Das gleiche gilt für die Interaktionsszenarios zur Dateneingabe (Forgiving Format, Checking for Correctness): ein Usability-Fehler, der die Nutzung der Anwendung verhindert, muss sofort behoben werden; zumal hier minimale Änderungen nötig sind.

2. Die Benutzer sind keine Techniker, sondern Berufskraftfahrer oder Rentner. Deshalb ist eine Anleitung wichtig, sie soll aber erst später integriert werden.

3. Die Realisierung des Interaktionsszenarios Using Applications Concurrently ist fremdbestimmt und erhält daher die niedrigste Priorität.

5.3.6.5 *Resultierende ASL-Tabelle*

Die Tabelle 79 illustriert die Ergebnisse der Evaluation (Spalte ASL0) und die Zielvorgaben für die nächsten Analysedurchgänge (ASL1 und ASL2). Alle Interaktionsszenarios bis auf Help und Using Applications Concurrently sollen in der nächsten Version der SA unterstützt werden. Die Hilfe soll erst in der darauf folgenden Iteration integriert worden sein. Das Interaktionsszenario „Using Applications Concurrently“ wird abschließend gesondert geprüft.

Szenarios	Begründung	Sensitive Entscheidungen	Unterstützung des Szenarios	Max. Einfluss auf Usability, Softwarearchitektur (ASL aktuell)	ASL0 (erste Analyse)	ASL1 (folgende Iteration)	ASL 2 (folgende Iteration)
Observing System State (System Feedback)	Walkthrough	S1 (k)	nein	2,2	2,2	0,0	0,0
Abbrechen (Canceling Commands)	Kontext: Walkthrough	S2 (k)	nein	2,2	2,2	0,0	0,0
Forgiving Format	Walkthrough	S3 (k)	nein	2,1	2,1	0,0	0,0
Checking for Correctness	Walkthrough	S4 (k)	nein	1,1	1,1	0,0	0,0
Context-based Workflow	Hauptszenario	S5, S6 (k)	teilweise (GPS-Ausfall probl.)	2,1	2,1	0,0	0,0
Using Applications Concurrently	Katalog	S7 (k)	nein	2,1	2,2	2,2	2/2 oder 0,0*
Maintaining Device Independence	Katalog	S8	ja	0,0	0,0	0,0	0,0
Progress Indication	Katalog	S9 (k)	nein	2,1	2,1	0,0	0,0
Familiar Appearance and/or Behaviour	Katalog	S10	ja	0,0	0,0	0,0	0,0
Verifying Resources	Katalog	S11	ja	0,0	0,0	0,0	0,0
Error Messages	Katalog	S12, S13, S14, S15	ja	0,0	0,0	0,0	0,0
Help	Katalog	S16	nein	2,2	2,2	2,2	0,0

Einfluss auf Usability
0... Die Anforderung wird umgesetzt.
1... Die Anforderung wird teilweise umgesetzt.
2... Die Anforderung wird nicht umgesetzt.
3... Die Anforderung wird nicht oder nur teilweise umgesetzt.

Einfluss auf Softwarearchitektur
0... Keine Modifikationen werden erwartet.
1... Einfache Modifikationen werden erwartet.
2... Komplexe Modifikationen werden erwartet.
3... Unvorhersehbare/unbekannte Modifikationen werden erwartet.

* abh. von Abrechnungsmodellen der Provider

Abbildung 79: Finale ASL-Tabelle für TrafficScanner

5.3.7 *Durchführung der Fallstudie TS: SATURN-Phase 5*

5.3.7.1 *Feedback*

Nach der Archivierung der Ergebnisse fand ein persönliches Gespräch über positives Feedback und Verbesserungsvorschläge statt.

- Die Analyse ist schnell und unkompliziert. Es gab sehr interessante Diskussionen und verwertbare Erkenntnisse für die aktuelle Projektarbeit. Die Ziele, die Softwarearchitektur zu beschreiben und das Reengineering vorzubereiten, wurden erreicht.
- Die Hinweise für die Usability-Evaluation sind auch für teaminterne Tests hilfreich. Die Szenarios können dort auch als Testszenarios zum Einsatz kommen.

Folgende Verbesserungsvorschläge wurden gegeben:

- Wie wichtig ein Szenario laut Anforderungsdokumentation oder Kundeneinschätzung ist, sollte ergänzt werden (Muss, Kann, Nicht notwendig). Beispielsweise ist die Hilfskomponente aus Auftraggebersicht bei TS nicht so wichtig, also ist es auch vertretbar, wenn sie nicht integriert wurde. Falls das nicht erwähnt wird, wirkt die Arbeitsleitung schlechter, obwohl das Gesamtergebnis gut ist.

5.3.7.2 *Pflege der Wissensbasis*

Folgende Szenarios wurden zur Recherche vorgeschlagen: Verifying Resources, Multi-Level Help. Das Szenario Workflow-Modell muss überarbeitet werden. Zudem fasst die folgende Tabelle zusammen, wie die Interaktionsszenarios verwendet wurden:

Interaktionsszenario	*Projekt*	*Bewertung*	*Häufigkeit (kumuliert)*
1. Feedback	SYM	(2, 2)	1
	TS	(2, 2)	2
4. Cancel	SYM	(2, 1)	1
	TS	(2, 2)	2
5. Forgiving Format	TS	(2, 1)	1
14. Checking for Correctness	TS	(1, 1)	1
21. Multi-Channel Access (Add new Input Device)	SYM	(2, 3)	1
24. Non-conflicting Application Concurrently	TS	(2, 1)	1
25. Device Independence	TS	(0, 0)	1
26. Recovering from Failure	SYM	(2, 2)	1
32. Progress Indication	TS	(2, 1)	1
34. Familiar Appearance and/or Behaviour	TS	(0, 0)	1
37. Verifying Resources	TS	(0, 0)	1
43. Error Messages	TS	(0, 0)	1
47. Multi-Level Help	TS	(2, 2)	1
49. Context-based Workflow (Context-aware Interaction)	TS	(2, 1)	1

Tabelle 36: Übersicht über die Verwendung von Interaktionsszenarios

5.3.8 *Beurteilung*

5.3.8.1 *Machbarkeit*

Bei der Analyse wurde eine sehr gut konstruierte Architektur ermittelt. Erstens werden vier Interaktionsszenarios (Maintaining Device Independence, Familiar Appearance and/or Behaviour, Verifying Resources und Error Messages) architektonisch durch eine gute Strukturierung und das Blackboard-Pattern unterstützt und erfordern damit keine Änderungen. Die Hauptinteraktion, die durch das Interaktionsszenario Context-based Workflow beschrieben wird, sowie die Interaktionsszenarios Progress Indication Forgiving Format und Checking for Correctness, werden teilweise unterstützt; eine kurze Prüfung konstruktiver Maßnahmen ermittelte voraussichtlich einfache Modifikationen, bei denen zwar Komponenten überarbeitet werden müssen, die grobe Struktur jedoch voraussichtlich erhalten bleibt. Die Interaktionsszenarios System Feedback, Abbrechen und Hilfe wurden schlecht bewertet und es sind voraussichtlich strukturelle Ände-

rungen notwendig. Wechselwirkungen mit den Qualitätsmerkmalen Modifizierbarkeit und Fehlertoleranz wurden offengelegt.

Es wurde mit der Methode SATURN ermittelt, dass die untersuchten Anforderungen, repräsentiert durch die Interaktionsszenarios, architektonisch teilweise unterstützt werden und weshalb dies so ist. Die Machbarkeit von SATURN ist für diese Fallstudie also bestätigt.

5.3.8.2 *Nützlichkeit*

Es stellt sich die Frage, ob der Architekt von TS es nützlich fand, diese Methode durchzuführen. Im Rahmen der Diagnose[5] wurden folgende Ziele festgelegt: Reengineering der Softwarearchitektur; Probleme ermitteln, die in neu zu erstellender Version behoben werden können (Abschnitt 5.3.1, S. 123).

Diese selbst gesetzten Ziele wurden erreicht: Die Architektur wurde so analysiert und untersucht, dass die Ergebnisse verwertbar sind. Die Interaktionsszenarios eignen sich außerdem für interne Tests. Ein weiteres Feedback war, dass die Analyse schnell und unkompliziert verlaufen ist. Aus Sicht des Architekten war die Architekturanalyse deshalb nützlich. (Abschnitt 5.3.7, S. 162).

5.3.8.3 *Verständnis*

VORGEHENSMODELL Die Arbeitsschritte wurden nach der kurzen Erklärung am Anfang verstanden. Auch in der Retrospektive sagt der Architekt, dass die Methode aus seiner Sicht logisch aufgebaut sei.

INTERAKTIONSSZENARIO Die Interaktionsszenarios wurden in einem Kurzvortrag präsentiert, Handzettel mit Tabellen und die ausgedruckten Interaktionsszenarios lagen vor. Während der zweiten Phase der Methode wurde der Architekt gebeten, jeweils die Inhalte der ausgewählten Interaktionsszenarios mit eigenen Worten zu umschreiben; es wurde deutlich, dass er sie korrekt verstanden hatte.

UNTERSTÜTZUNG DURCH HILFSMITTEL Zusätzlich zu den Interaktionsszenarios waren Evaluationsformulare mit einem Tabellenverarbeitungsprogramm vorbereitet worden; sie wurden während der Analyse ausgefüllt.

5.3.9 *Reflexion*

5.3.9.1 *Aufwand*

Der Zeitaufwand für die Analyse von TrafficScanner betrug 2 Personentage für zwei Personen, also vier Personentage. Es fanden vier vierstündige Termine innerhalb eines Monats in einem Leipziger Unternehmen statt. Die Aufwände des Forschers wurden nicht mitberechnet, da die Beobachtung nicht zur Methode zählt. Da zwei Personen die Methode durchführten (Analyst und Architekt), betrug der Gesamtaufwand für die Fallstudie vier Personentage.

Gerade der Aufwand der ersten Phase von SATURN war größer als erwartet. TrafficScanner wurde mit Extreme Programming erstellt und es gab keine aktuelle Architekturdokumentation, die mehr als nur Komponenten und grobe Aufgaben enthielt. Um den tatsächlichen Zeitaufwand für die Methode SATURN gut überschaubar zu halten, sollen Art und Umfang der Architekturbeschreibung vorher überprüft werden.

Abgesehen davon ist der Architekt die wichtigste Informationsquelle, denn während der Evaluation der Interaktionsszenarios werden weitere Informationen benötigt, die

5 siehe Canonical-Action-Research-Zyklus in Abschnitt 5.1.3 ab S. 86

entweder noch nicht als Architekturentscheidungen notiert oder noch nicht beachtet wurden.

5.3.9.2 *Aufbau und Inhalte der Methode*

Bevor die Methode SATURN durchgeführt werden kann, soll eine Prüfung der Architekturbeschreibung am Anfang feststellen, ob diese ausführlich genug beschrieben ist. Dazu werden Kriterien basierend auf den Fallstudien und wissenschaftlichen Quellen über Architekturbeschreibungen aufgestellt.

In der Fallstudie TrafficScanner findet in der ersten Phase ein Mapping von Interaktionen auf die Interaktionsklassen von CASSINI statt. Im Anschluss daran werden in der zweiten Phase Interaktionsszenarios aus diesem Katalog ausgesucht. Da die erste Aufgabe der zweiten sehr ähnelt und auf das Abstraktionsniveau der Phase 2 vorgreift, ist sie zum einen doppelt und zum anderen zu detailliert, weshalb sie gestrichen wird.

In der vierten Phase der Fallstudie TrafficScanner werden die Interaktionsszenarios vorm Hintergrund des Nutzungskontextes abschließend erneut priorisiert. Die Tabelle am Beginn der vierten Phase dient aber dazu, die Arbeitsweise der Phase 3 zusammenzufassen und auf diese zurückzublicken, um den Weg für eine neue Perspektive auf die Arbeitsergebnisse frei zu machen. Diese Phase beschäftigt sich besonders mit dem Auffinden von fachlich zusammenhängenden Architekturentscheidungsclustern und ihrer Beschreibung. Konstruktive Arbeitsschritte der Softwareentwicklung sollen empfohlen, aber nicht konkretisiert werden, denn diese Arbeit ist auf die Analyse spezialisiert. Vor diesem Hintergrund ist eine abschließende Betrachtung des Nutzungskontexts und des ASL nicht nötig.

5.3.9.3 *Wechselwirkungen*

Zusammenfassend wurden folgende Austauschbeziehungen mit anderen Qualitätsmerkmalen ermittelt.

- Modifizierbarkeit: Durch eine gute Separation der bestehenden Komponenten können grundsätzlich andere Methoden zur Ortsbestimmung (S14), die gewünschte Funktionalität Telefonieren (S7) integriert und externe Komponenten leicht ausgetauscht (S8) werden.
- Fehlertoleranz: Prinzipiell werden vorab definierte Benutzungsfehler und Systemfehler erkannt und behandelt (S12, S13, S14, S15). Allerdings ist die Abhängigkeit von den fünf aktuellen und nicht zwischengespeicherten Positionsdaten sehr kritisch (S14), da die Anwendung schon bei einem kurzen Ausfall eines angeschlossenen Systems oder bei schlechter Netzabdeckung nicht mehr funktioniert. Zudem ist es nicht möglich, Prozesse abzubrechen (S2), so dass die Anwendung geschlossen werden muss, um fehlerhafte oder unerwünschte Prozesse zu beenden. Nutzer können sich außerdem nicht mit korrekt eingegebenen IDs anmelden (S3, S4), so dass sie die Anwendung gar nicht starten können.

Die aktuellen Architekturentscheidungen unterstützen sowohl Modifikation als auch Usability. Das derzeitige Maß an Fehlertoleranz behindert die Usability jedoch deutlich.

5.4 ZUSAMMENFASSUNG

In diesem Kapitel wurden die zwei Fallstudien dokumentiert, die dazu erstellt wurden, um die Methode SATURN auf Machbarkeit zu prüfen.

Die Vorgehensweise von SATURN wurde logisch nachvollziehbar verstanden, ebenso die Interaktionsszenarios. Es wurde ebenso erforscht und erprobt, wie eine Softwarearchitekturanalyse und eine User-Evaluation miteinander verbunden werden können, damit die Usability ganzheitlich betrachtet werden kann und ersichtlich wird, wie ihre Resultate miteinander in Beziehung stehen. Insbesondere wurde vorgeschlagen, Usability-Anforderungen als eine gemeinsame Basis für die Softwarearchitekturanalyse (eine Inspektionsmethode) und eine User-Evaluation (ein Benutzertest) zu verwenden und deren Ergebnisse in Form von Usability-Problemen und Architektur-Problemen aufeinander abzubilden und die Resultate als Usability-Probleme zu diskutieren.

Nach den Fallstudien wurde die Methode SATURN wie folgt verbessert: die Anleitung wurde gekürzt, Tabellen und Checklisten wurden ergänzt, der Interaktionsszenariokatalog wurde erweitert, ins Deutsche übersetzt und redaktionell bearbeitet, schließlich wurde das Ergebnis der Betrachtung von Qualitätsmodellen für die vierte Phase als Liste typischer Qualitätsmerkmale ergänzt.

Ein vormals obligatorischer Pattern-Workshop wurde gestrichen, da die Pflege einer eigenen Pattern-Bibliothek optional ist und einen anderen Themenkomplex darstellt. Es reicht zudem aus, vorhandene Patterns lesen zu können, die auch über Bücher, wissenschaftliche Veröffentlichung oder Online-Datenbanken abrufbar sind.

Auch die Beschreibung der Softwarearchitektur wurde ausgelagert: sie ist nun vor der Analyse zu erstellen. Wie genau das geschehen soll und welche Informationsquellen notwendig sind, wurde basierend auf den Erfahrungen der zwei Fallstudien vorgegeben. Es wurde deutlich, dass die Architektur soweit beschrieben sein muss, dass die Möglichkeiten und Grenzen von Nutzer-System-Interaktionen erkennbar sind. Aussagen über die Usability wären schwer nachweisbar gewesen, wenn beispielsweise nur Datenmodelle über die fachliche Architektur vorhanden gewesen wären. Die Architekturebenen 1 und 2 enthielten die Aussagen über beispielsweise die Verteilung der Kommunikation (Client/Server, Peer-to-Peer, MVC, PAC), Kommunikationsprotokolle und die Definition von Schnittstellen.

Hinsichtlich der Dokumentation der SA ist zu beachten, dass eine solche Analyse immer eine tiefe Analyse ist, die offenlegt, wie eine bestimmte Nutzer-System-Interaktion realisiert wird oder werden soll. Bei szenario-basierten Softwarearchitekturanalysemethoden sind die Architekten die wichtigste Informationsquelle der Anwendung. Sie müssen, sofern das zu diesem Zeitpunkt noch nicht geschehen ist, Aussagen über die Umsetzung einzelner Interaktionen treffen können.

Es folgt das Kapitel 6 Validation (ab S. 167). Es prüft, inwiefern selbstgesetzte Validationsziele erreicht werden.

DANKSAGUNGEN

Die wissenschaftliche Studie und die Kombination der User Evaluation und SATURN erschien im *Journal of Software and Systems (JSS)* [BGG10]. Die Industriefallstudie erschien in den Proceedings der *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)* [BG10a]. Vielen Dank an alle Gutachter und Kommentatoren. Herzlichen Dank an Falk Rehwagen und Kollegen, Thomas Grill, Stefan Seitz und Volker Gruhn.

6 VALIDATION DER METHODE SATURN

Zuerst werden Validationsziele vorgestellt (Abschnitt 6.1). Sie werden danach genauer betrachtet: konstruktive Validität (Abschnitt 6.2), interne Validität (Abschnitt 6.3) und externe Validität (Abschnitt 6.4). Abschnitt 6.5 thematisiert, inwiefern mit SATURN in den Fallstudien und im Vergleich zu den betrachteten früheren Methoden nützlich ist. Abschnitt 6.6 fasst das Kapitel zusammen.

6.1 VALIDATIONSZIELE

In der Kritischen Theorie sind sowohl Problem als auch Prozess zur Bewältigung des Problems bedeutend [ESSD07]. Eine Methode ist somit dann valide, wenn sie das Problem bewältigt und den Prozess zur Bewältigung des Problems für bestimmte Individuen und Gruppen verbessert. Diese Validation muss also folgende zwei Fragen thematisieren:

1. Validation der Methode: Ist die Methode SATURN geeignet, um von Usability-Anforderungen ausgehend die Software-Architektur zu analysieren und Probleme zu ermitteln? (Abschnitte 6.2, 6.3, 6.4)
2. Nützlichkeit: Welchen Nutzen haben Architekturanalysen allgemein? Inwiefern waren die Fallstudien aus Sicht der Personen nützlich, die sie durchgeführt haben? Wie nützlich ist SATURN verglichen mit früheren Arbeiten? Wie behandelt diese Methode allgemeine Probleme zur Nützlichkeit szenario-basierter Softwarearchitekturanalysen? (Abschnitt 6.5)

Damit wissenschaftliche Erkenntnisse als gesichert und gültig akzeptiert werden können, muss ihre Validität überprüft werden. Nach Easterbrook [ESSD07] sind vier Aspekte von Validität relevant: die konstruktive, interne und externe Validität sowie die empirische Reliabilität. Konstruktive Validität wird erreicht, wenn gemessen wurde, was gemessen werden sollte. Interne Validität wird erreicht, wenn Resultate theoretisch nachvollziehbar von den Daten abgeleitet wurden und wenn anhand der Ergebnisse der Fallstudien keine Invalidation stattfindet. Zudem sollen das Ergebnis verwischende Variablen bekannt sein und Maßnahmen zur Verringerung ihres Einflusses ergriffen worden sein. Externe Validität erfordert es, dass die Resultate den Anspruch der Generalisierbarkeit unterstützen, d.h. dass die Resultate zu einem Prozess der analytischen Verallgemeinerung beitragen. Empirische Reliabilität gilt für Forschung in einem längerfristigen Reifeprozess und beschäftigt sich mit der Frage, ob Aussagen über Generalisierbarkeit getroffen werden können. Hierbei ist der Begriff der empirischen Induktion wichtig: die Beweiskraft ist dann darauf aufgebaut, dass nachfolgende Fallstudien eine Theorie ebenso unterstützen wie vorhergehende Fallstudien. [ESSD07]

Was valide ist, hängt generell vom Ziel und vom Anspruch eines Forschungsprojektes ab. Wie Tabelle 37 (S. 168) zeigt, kann der Forschungsprozess in drei aufeinander aufbauende Stufen unterteilt werden, welche unterschiedliche Forschungsziele verfolgen, die mit unterschiedlichen wissenschaftlichen Methoden erreicht werden und den jeweiligen Validationsanforderungen genügen müssen. [BM85]

	Stufe 1	*Stufe 2*	*Stufe 3*
Ziele	Definitionen und logische Zusammenhänge	Methode erstellen und deren Durchführbarkeit prüfen	Verallgemeinerung der Ergebnisse
Methodik	Literaturstudie	Anwendungsnahe Forschung, qualitativ	Quantitative Forschung
Reife	Konzeption: Begriffe, theoretischer Rahmen	Praktische Machbarkeit: Methode	Reifeprozess der Methode und des Wissens
Validationsziele	konstruktive Validität: Gültigkeit der Begriffe und ihrer Beziehungen	konstruktive, interne und externe Validität	konstruktive, interne, externe Validität und empirische Reliabilität
Ziele dieser Arbeit	✓	✓	zukünftiger Forschung vorbehalten

Tabelle 37: Aspekte von Validität und Zielvorgabe dieses Forschungsprojektes [BM85]

In Stufe 1 soll mittels Literaturstudien ein konstruktiv valider Rahmen für die Forschungsarbeit erstellt werden. Stufe 2 verwendet qualitative, anwendungsnahe Forschung und will die Machbarkeit einer Forschung absichern, die zudem nachweisbar intern und extern valide sein soll. Erst in Stufe 3 kommt die empirische Reliabilität hinzu, bei der die Forschung durch einen Reifeprozess nach Allgemeingültigkeit strebt.

In dieser Forschungsarbeit wird eine neue Softwarearchitekturanalysemethode konstruiert und erprobt, der Reifeprozess soll späteren Forschungsprojekten überlassen sein, in denen diese Methode mehr und mehr verwendet und verbessert wird. Dieser Reifeprozess wird also von diesem Forschungsprojekt ausgeschlossen, er soll aber methodisch unterstützt werden. Dieses Forschungsprojekt bezieht sich damit auf die Stufen 1 und 2 des dargestellten Forschungsprozesses.

Die Methode SATURN kann also als valide bezeichnet werden, wenn folgende Validationsziele der Stufen 1 und 2 erreicht werden:

- Stufe 1 – Konstruktive Validität: Wurde gemessen, was gemessen werden sollte? Theoretische Elemente und die Beziehungen zwischen ihnen sollen wissenschaftlich untermauert sein. Ergebnisse sollen wissenschaftlich korrekt in beobachtbare Maße umgewandelt werden; ebenso sollen Vorgaben für die Bewertung und verwendete Skalen wissenschaftlich korrekt, logisch nachvollziehbar und somit valide sein.

- Stufe 2 – Interne Validität: Sind Resultate theoretisch nachvollziehbar von den Daten abgeleitet? Die Resultate sollen theoretisch nachvollziehbar von ermittelten Daten abgeleitet werden. Auch anhand der Fallstudienergebnisse soll keine Invalidation stattfinden. Das Ergebnis verwischende Variablen sollen bekannt sein, Maßnahmen zur Verringerung ihres Einflusses sollen ergriffen worden sein.

- Stufe 2 – Externe Validität: Unterstützen die Resultate prinzipiell den Anspruch der Generalisierbarkeit? Die Resultate tragen zu einem Prozess der analytischen Verallgemeinerung bei.

Im Sinne eines Kosten-Nutzen-Vergleichs werden außerdem folgende Ziele für die Nützlichkeit gesetzt:

- Die Personen, die die Fallstudien durchgeführt haben, sollen die Analyse als nützlich erachten.
- Der Aufwand der neuen Methode soll geringer sein als der Aufwand der betrachteten früheren Arbeiten. Der Nutzen durch die Aussagekraft der Ergebnisse soll höher sein als die Aussagekraft der betrachteten früheren Arbeiten.
- Typische Probleme szenario-basierter Softwarearchitekturanalysemethoden sollen bekannt sein. Maßnahmen zur Verringerung ihres Einflusses sollen ergriffen worden sein.

Im Folgenden werden diese einzelnen Aspekte der Validität nacheinander abgehandelt.

6.2 KONSTRUKTIVE VALIDITÄT

6.2.1 *Forschungsmethodik*

Typische Forschungsmethoden beim Konstruieren von Methoden sind Fallstudien und Action Research, die durch Literaturstudien ergänzt werden [BWHW05]; in dem Sinne wurden frühere szenario-basierte Softwarearchitekturanalysemethoden generell mit Fallstudien validiert [CKK02, BG04, BLBV04].

Analog dazu wurde die Methode SATURN basierend auf Literaturstudien konstruiert. Hinzugezogen wurde Literatur zu früheren Arbeiten (siehe Kapitel 2), zum Usability Engineering, Patterns (siehe Kapitel 3 und 4). Mit Hilfe der Aktionsforschung (siehe Kapitel 5, ab S. 5) wurde die theoretisch konstruierte Methode SATURN in zwei Fallstudien erprobt und verfeinert.

6.2.2 *Definitionen der einzelnen theoretischen Elemente*

In SATURN werden als theoretische Elemente die Begriffe Qualität, Usability, Softwarearchitektur, Architekturentscheidungen, Mobilität, Nutzungskontext und Interaktionsszenarios miteinander verbunden. Diese Begriffe sind im Rahmen des Metamodells der Methode SATURN in den Abschnitten 4.2.1.1, S. 62 und 4.2.1.2, S. 63 vorzufinden; sie sind fachlich sinnvoll und korrekt definiert.

6.2.3 *Zusammenhänge zwischen den theoretischen Elementen*

Mit der Methode SATURN wird analysiert, inwiefern bestimmte Interaktionsszenarios einer mobilen Anwendung architektonisch unterstützt werden. Basierend auf dem Nutzungskontext werden Interaktionsszenarios ausgewählt oder erstellt. Der Nutzungskontext bestimmt Rahmenbedingungen für die Usability, von denen Interaktionsszenarios abgeleitet werden können, zum Beispiel von den Aufgaben der Nutzer, ihren Endgerätetypen und den Umgebungen, in denen sie mit der Anwendung interagieren wollen. Interaktionsszenarios zeigen also Anwendungsfälle mit der Anwendung auf, die von Usability-Rahmenbedingungen abgeleitet wurden. Beim Evaluieren von Interaktionsszenarios werden architektonische Elemente ermittelt, durch deren Kollaborationen die Response eines Interaktionsszenarios umgesetzt oder nicht umgesetzt werden kann. Diese Architekturentscheidungen werden formuliert und bewertet.

Zuerst wird der Einfluss einer Architekturentscheidung auf das Interaktionsszenario geprüft: ist diese Architekturentscheidung wesentlich, um die Response des betreffenden Interaktionsszenarios zu erreichen? Wenn nein, wird die nächste Entscheidung betrachtet. Wenn ja, muss die Frage beantwortet werden, ob die Architekturentscheidung die Response fördert oder behindert (siehe Tabelle 4.1.7, S. 56). Falls letzteres der Fall ist, sollen alternative Lösungsvorschläge genannt und grob geschätzt werden, wie aufwendig Änderungen an der Architektur sein können (siehe Abschnitt 4.1.8, S. 57). Um Probleme gut sichtbar zu machen, bestimmen die schlechtesten Bewertungen der Architekturentscheidungen zu Usability und notwendigen Architekturänderungen die Gesamtbewertung eines Interaktionsszenarios.

Die architektonische Unterstützung von Usability wird in der Architektur-Support-Level-Tabelle zusammengefasst; die Bewertungen jedes analysierten Interaktionsszenarios auf die Softwarearchitektur werden aufzeigt. In dieser Tabelle kann durch die Bestimmung von Bewertungszielen auch festgelegt werden, welche Interaktionsszenarios in einer bestimmten Folgeversion der Softwarearchitektur zu berücksichtigen sind.

In SATURN wird sowohl durch das Vorgehensmodell als auch durch Vorlagen explizit vorgegeben, welche Daten zu erheben, zu dokumentieren und weiterzuverwenden sind. Die logische Abfolge der Analyse und Bewertung der architektonischen Unterstützung von Usability wird also durch das Vorgehensmodell angeleitet.

6.2.4 *Vorgaben für die Bewertungen*

Dieser Abschnitt beschäftigt sich damit, ob Ergebnisse in beobachtbare Maße umgewandelt wurden. Dafür sollen Vorgaben für die Bewertung wissenschaftlich korrekt und damit valide sein. Zudem sollen auch die ausgewählten Skalen logisch nachvollziehbar, korrekt und damit valide sein.

Die Bewertung des voraussichtlichen Einflusses von Architekturentscheidungen auf Usability basiert darauf, dass ein Interaktionsszenario als Anforderung betrachtet wird. Grundsätzlich wird der Einfluss der Architekturentscheidungen auf Usability über die einfach zu beantwortende Frage bewertet, ob die Response eines Interaktionsszenarios komplett (Typ 0), teilweise (Typ 1) oder nicht (Typ 2) architektonisch umgesetzt werden kann. Welche alternativen Varianten möglich wären und warum gerade diese ausgewählt wurde, beschreibt Abschnitt 4.1.7, S. 56.

Die Bewertung des potenziellen Änderungsaufwandes der Architektur basiert auf der verwendeten Architekturdefinition, die Architekturentscheidungen akzentuiert, und der betrachteten Architekturebene. Hier werden die optionalen alternativen Architekturentscheidungen grob nach ihrem Änderungsaufwand typisiert. Typ 0 für keine Änderungen vergeben, Typ 1 für Modifikationen innerhalb einer Komponente, Typ 2 für Änderungen für komplexe Modifikationen, die durch das Ergänzen oder Entfernen von einer oder mehreren Komponente(n) entstehen, da dann meist auch andere bestehende Komponenten modifiziert werden müssen. Typ 3 umfasst also unvorhersehbare oder unbekannte Modifikationen (offene Fragen). Details dazu befinden sich in Abschnitt 4.1.8, S. 57.

Als Skalen für den Einfluss auf Usability bzw. auf die Softwarearchitektur wurde die Ordinalskala gewählt. Diese Skala eignet sich, da sie die typische innere Ordnung der Daten abbildet: die Elemente in den Kategorien haben bestimmte Eigenschaften, die Abstände zwischen den Kategorien sind undefinierbar.

6.2.5 *Wahl der Skalen*

Unzulässige Änderungen der Skala werden nicht vorgenommen; es werden also keine Durchschnittswerte gebildet. Die Gesamtbewertung eines Interaktionsszenarios (Ein-

fluss auf Usability, Modifikationsaufwand) wird durch die schlechtesten Bewertungen aller Architekturentscheidungen eines Interaktionsszenarios bestimmt.

Ein Durchschnittswert, der für die Ordinalskala generell nicht gebildet werden sollte, aber häufig gebildet wird (z.B. Notendurchschnitt), verschleiert Probleme: viele gute Bewertungen und wenige, aber sehr kritische Bewertungen, führen zu einer guten Gesamtbewertung. Das widerspräche dem Hauptziel von SATURN, Usability-Probleme und notwendige Architekturänderungen offenzulegen. Die Bewertung eines Interaktionsszenarios zeigt die schlechteste Bewertung des Einflusses auf die Usability und die schlechteste Bewertung des Änderungsaufwandes der Architektur; das Architektur Support Level zeigt keine Durchschnittswerte, sondern einzelne Gesamtbewertungen von Interaktionsszenarios. Diese Art der Bewertung behindert die interessengeleitete Manipulation, bei der eine große Menge positiver Ergebnisse den Durchschnitt positiv beeinflusst. Diese Art von Manipulation würde sich auf die Effizienz der Methode auswirken, da nicht nur die Probleme, sondern insbesondere gute Ergebnisse in den Vordergrund gedrängt werden würden. Deshalb ist die Beibehaltung der Ordinalskala auch aus fachlicher Sicht sinnvoll, um den Einfluss eines Interaktionsszenarios zu verdeutlichen.

6.2.6 *Zusammenfassung zur konstruktiven Validität*

Es wurde gezeigt, dass die einzelnen theoretischen Elemente der Methode SATURN fachlich sinnvoll und korrekt definiert sind; dass die Beziehungen zwischen ihnen nachvollziehbar sind. Architekturentscheidungen werden dahingehend überprüft, ob sie mit einem Interaktionsszenario konform sind. Da eine Architekturentscheidung die Softwarearchitektur operationalisiert und ein Interaktionsszenario Rahmenbedingungen für die Usability operationalisiert, wird gemessen, was gemessen werden sollte: der architektonische Einfluss auf die Usability. Die Bewertungsvorgaben für Usability (Response) sowie Architekturänderungen (strukturelle Änderungen, Verhaltensänderungen) und gewählte Skalen (Ordinalskala) sind korrekt und nachvollziehbar. Die Konstruktion der Methode SATURN ist also gültig.

6.3 INTERNE VALIDITÄT

6.3.1 *Theoretische Nachvollziehbarkeit der Resultate*

Der Nutzungskontext beschreibt Usability-Rahmenbedingungen 4.1.2 (S. 48). Von ihm werden Interaktionsszenarios abgeleitet. Diese Interaktionsszenarios beschreiben, wie die mobile Anwendung oder ein Teil der mobilen Anwendung auf eine nutzer- oder systeminitiierte Interaktion reagieren sollen; beschrieben werden die gewünschten Reaktionen der Response (funktional) und Response-Prüfung (qualitativ) (Kapitel 3, S. 27 ff.).

Während der Analyse werden Teilnutzfälle des betreffenden Interaktionsszenarios untersucht. Entsprechend den Sichtenmodellen auf Softwarearchitekturen (Abschnitt 4.2.1.4, S. 64 ff.) werden architektonische Elemente und ihr Verhalten zu einem Teilnutzfall ermittelt (Abschnitt 4.2.5, S. 69 ff.).

Die ermittelten architektonischen Elemente und ihr Verhalten werden als Architekturentscheidungen formuliert und es wird bewertet, inwiefern sie mit dem Interaktionsszenario konform sind, d. h. mit der Response und der Response-Prüfung (siehe Abschnitt 4.2.5, S. 69 ff.).

Wenn die Architekturentscheidungen nicht mit dem Interaktionsszenario konform sind, werden alternative Architekturentscheidungen vorgeschlagen; und es wird grob bestimmt, wie aufwendig diese Änderungen der aktuellen Architektur sein können.

Unterschieden werden Verhaltensänderungen der architektonischen Elemente (Typ 1), strukturelle Änderungen (Typ 2) und unvorhersehbare Änderungen (Typ 3).

Die Gesamtbewertung des jeweiligen Interaktionsszenarios zeigt den größten negativen Einfluss auf die Response und Response-Prüfung und den Aufwand der Architekturänderungen.

Um den Argumentationspfad aufzuzeigen, ist eine detaillierte Darstellung notwendig, die den Zusammenhang zwischen Nutzungskontextinformationen, Interaktionsszenarios, Nutzfällen, szenario-beeinflussenden Architekturentscheidungen und Modifikationsvorschlägen übersichtlich darstellt. Das Schema in Tabelle 6.3.1 orientiert sich an den Analyseformularen und bezieht sowohl die Ergebnisse der Phase 3 (Evaluation der Interaktionsszenarios) als auch die Ergebnisse der Phase 4 (Interpretieren der Resultate) mit ein. Für die interne Validität steht im Mittelpunkt, dass die Ergebnisse nachvollziehbar von den Daten abgeleitet wurden und dass anhand der Fallstudienergebnisse keine Invalidation stattfand. Die ermittelten Architekturentscheidungen zu den evaluierten Interaktionsszenarios sind nach dem folgenden Schema tabellarisch aufbereitet.

Projekt	Projektname
Nutzungskontext	Rahmenbedingungen der Nutzung, u.a. wichtigste Aufgaben der Nutzer (Zweck der Anwendung, Abgrenzung zur Konkurrenz); Phase 1
Interaktionsklasse(n)	Mapping konkreter Aufgaben auf Interaktionsklassen (fachliche Sortierung der Interaktionsszenarios); Phasen 1 und 2
Interaktionsszenario	Nummer eines verwendeten, schon vorhandenen Interaktionsszenarios, das durch die Literaturstudie ermittelt wurde; Phase 2
Usability-Anforderung	Beschreibung von Stimulus und Response; Phase 2
Use Case(s)	Spezialisierung des Interaktionsszenarios in genauere Use Cases; Phase 3
Architektur-entscheidung(en)	Existenzentscheidungen und inwiefern sie es ver- oder behindern, das Interaktionsszenario später mit der Anwendung durchzuführen; Phase 3
Quelle(n)	schon dokumentierte Architekturentscheidungen oder noch nicht dokumentierte, die erst durch die Methode erhoben wurden; Phase 3
Modifikation(en)	Änderungsvorschläge, um ermittelte Probleme zu beheben; Phase 3 (optional)
Gesamtbewertung	architektonischer Einfluss auf die Durchführbarkeit des Interaktionsszenarios durch aktuelle Architekturentscheidungen und Änderungsaufwand bezüglich alternativ vorgeschlagener Architekturentscheidungen; Phase 3

Tabelle 38: Schema für Argumentationspfade

6.3.2 *Ergebnisse der Fallstudien*

Basierend darauf können folgende separate Argumentationspfade für jedes untersuchte Interaktionsszenario erstellt werden.

Projekt	SYM
Nutzungskontext	Aufgabe (Walkthrough)
Interaktionsklasse(n)	keine (bei der Fallstudie SYM gab es noch keine Interaktionsklassen)
Interaktionsszenario	*4. Abbrechen*
Usability-Anforderung	Stimulus – „Benutzer haben eine Operation gestartet und möchten sie stoppen und widerrufen.“, Response – „Das System beendet die Ausführung, sobald der Nutzer die Abbrechen-Funktion aktiviert. Der vor dem Starten des Vorgangs bestehende Systemstatus wird wieder hergestellt.“
Use Case(s)	Navigieren (Hauptinteraktion)
Entscheidung(en)	S1 – Existenz spezieller Distributionskomponente (ok), S2 – Transformation durch Anwendungscontroller (ok), S3 – Controller erlaubt keine direkte Manipulation (Behinderung der Response Typ 2)
Quelle(n)	S1 – Dokumentation UML-Diagramm S2 – Dokumentation UML-Diagramm, Befragung des Architekten S3 – Befragung des Architekten
Modifikation(en)	S3 – Verwendung einer Queue mit Prioritäten (SA-Sensitivität Typ 1)
Gesamtbewertung	(2, 1)

Tabelle 39: Argumentationspfad des Interaktionsszenarios Abbrechen in Fallstudie SYM

*P*rojekt	SYM
Nutzungskontext	Aufgabe (Brainstorming)
Interaktionsklasse(n)	keine (bei der Fallstudie SYM gab es noch keine Interaktionsklassen)
Interaktionsszenario	*21. Multi-Channel Access*
Usability-Anforderung	Stimulus – „Benutzer möchten Zugang zum System mithilfe einer Hardware ihrer Wahl (und damit Zugang für die jeweiligen Eingabe- und Ausgabemöglichkeiten).“, Response – „Das Eingabe/Ausgabe-Gerät eines Benutzers wird mit der Hardware des Systems verbunden und funktioniert mit der Anwendung.“
Use Case(s)	Neues Endgerät ergänzen (z. B. iPhone), weitere Controller ergänzen (z. B. Firefox auf weiterem PC)
Entscheidung(en)	S4 – Nichtexistenz einer Regelung zum Ergänzen neuer Endgeräte, z.B. iPhone (Behinderung der Response Typ 2), S5 – Nichtexistenz einer Regelung zum Ergänzen neuer Anwendungscontroller (Behinderung der Response Typ 2)
Quelle(n)	S4 – Befragung des Architekten *Gesamtbewertung* S5 – Dokumentation UML-Diagramm, Befragung des Architekten
Modifikation(en)	S4 – Komponenten für neue Eingabegeräte und die Bearbeitung der jeweiligen Rohdaten der Bewegungssensoren, Verbindungskontrolle (SA-Sensitivität Typ 3), S5 – Umstellung von TCP/IP auf UDP oder Mix von TCP/IP und UDP (SA-Sensitivität Typ 2), neue Komponente zur Verteilung von Events auf dem Server (SA-Sensitivität Typ 3)
*B*ewertung	(2,3)

Tabelle 40: Argumentationspfad des Interaktionsszenarios Multi-Channel Access in Fallstudie SYM

Projekt	SYM
Nutzungskontext	Aufgabe (nach Walkthrough)
Interaktionsklasse(n)	keine (bei der Fallstudie SYM gab es noch keine Interaktionsklassen)
Interaktionsszenario	*1. System-Feedback*
Usability-Anforderung	Stimulus – „Benutzer möchten über Status, Aktivitäten und Veränderungen des Systems informiert bleiben.“, Response – „Das System liest die vom Nutzer angeforderten Daten aus und zeigt sie entsprechend der menschlichen Bedürfnisse und Fähigkeiten an.“
Use Case(s)	Interaktion mit Google Earth
Entscheidung(en)	S6 – GUI zeigt vom Server erkannte Interaktion an (ok), S7 – Controller zeigt nicht die bearbeitete Interaktion an (Behinderung der Response Typ 2), S8 – Verwendung von TCP/IP als Netzwerkprotokoll (Behinderung der Response Typ 2)
Quelle(n)	S6 – Befragung des Architekten, Walkthrough durch das User Interface S7 – Befragung des Architekten, Walkthrough durch das User Interface S8 – Dokumentation hinsichtlich verwendeter Protokolle
Modifikation(en)	S7 – Ergänzung eines Status und einer Statusanzeige im UI (SA-Sensitivität Typ 1), S7 – Änderung eines Software-Interfaces und der erbenden Controller (SA-Sensitivität Typ 1), S8 – Umstellung von TCP/IP auf UDP oder Mix von TCP/IP und UDP (SA-Sensitivität Typ 2)
Gesamtbewertung	(2, 2)

Tabelle 41: Argumentationspfad des Interaktionsszenarios System-Feedback in Fallstudie SYM

Projekt	SYM
Nutzungskontext	Benutzer und Umgebung (Besucher einer Ausstellung)
Interaktionsklasse(n)	keine (bei der Fallstudie SYM gab es noch keine Interaktionsklassen)
Interaktionsszenario	*26. Wiederherstellung nach Betriebsausfall*
Usability-Anforderung	Stimulus – „Das System arbeitet fehlerhaft oder stürzt ab.“, Response – „Das System minimiert die Menge an aufgrund des Defekts verloren gehenden Daten oder stellt Nutzern Mittel zur Verfügung, um den Schaden gering zu halten.“
Use Case(s)	Client verliert Verbindung
Entscheidung(en)	S9 – Nichtexistenz Monitoring oder Management der Konnektivität in beide Richtungen: Eingabe und Ausgabe (Behinderung der Response Typ 2)
Quelle(n)	S9 – Dokumentation UML-Diagramm, Befragung des Architekten
Modifikation(en)	S9 – Monitoring und automatisches Konnektieren der Verbindung zu mobilen Endgeräten (SA-Sensitivität Typ 2)
Gesamtbewertung	(2, 2)

Tabelle 42: Argumentationspfad des Interaktionsszenarios Wiederherstellung nach Betriebsausfall in Fallstudie SYM

Projekt	TS
Nutzungskontext	Hauptverwendungszweck, Aufgabe
Interaktionsklasse(n)	Navigation, Browsen, Auswählen; Strukturieren und Anzeigen von Content, Informationen, Daten; Orientierung
Interaktionsszenario	*1. System-Feedback*
Usability-Anforderung	Stimulus – „Benutzer möchten über Status, Aktivitäten und Veränderungen des Systems informiert bleiben.“, Response – „Das System liest die vom Nutzer angeforderten Daten aus und zeigt sie entsprechend der menschlichen Bedürfnisse und Fähigkeiten an.“
Use Case(s)	Staumeldung (Hauptszenario)
Entscheidung(en)	S1 – Nichtexistenz eines Status zum Versenden oder eine Meldungswarteschlange (Behinderung der Response Typ 2)
Quelle(n)	S1 – Aus Quelltext generierte Dokumentation, Befragung des Architekten
Modifikation(en)	S1 – Ergänzung einer Meldungswarteschlange (SA-Sensitivität Typ 2) inkl. Ergänzung eines Status „sende gerade“ und der Anzeige des Status im UI
Gesamtbewertung	(2, 2)

Tabelle 43: Argumentationspfad des Interaktionsszenarios System-Feedback in Fallstudie TS

Projekt	TS
Nutzungskontext	Aufgabe
Interaktionsklasse(n)	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Interaktionsszenario	*4. Abbrechen*
Usability-Anforderung	Stimulus – „Benutzer haben eine Operation gestartet und möchten sie stoppen und widerrufen.“, Response – „Das System beendet die Ausführung, sobald der Nutzer die Abbrechen-Funktion aktiviert. Der vor dem Starten des Vorgangs bestehende Systemstatus wird wieder hergestellt.“
Use Case(s)	Abbrechen des Sendens
Entscheidung(en)	S2 – Cancel wird nicht architektonisch unterstützt (Behinderung der Response Typ 2)
Quelle(n)	S2 – Aus Quelltext generierte Dokumentation (Schnittstellen zum System und deren Implementierung), Befragung des Architekten
Modifikation(en)	S2 – Querschnittsaufgabe, Kooperation mit Plattform möglich (SA-Sensitivität Typ 2)
Gesamtbewertung	(2, 2)

Tabelle 44: Argumentationspfad des Interaktionsszenarios Abbrechen in Fallstudie TS

*P*rojekt	TS
Nutzungskontext	Aufgabe
Interaktionsklasse(n)	Erfassung und Behandlung von Benutzereingaben;
*S*zenario	*5. Nachsichtiges Format*
Usability-Anforderung	Stimulus: „Das System bietet den Nutzern ein Formular zur Datenabfrage an.“, Response: „Nutzer geben die Daten in ein Feld ein und müssen dabei kein besonderes Format und keine besondere Syntax berücksichtigen. Das System interpretiert die Eingaben selbständig.“
Use Case(s)	Login
Entscheidung(en)	S3 – Feedback bei Fehler ist Eingabe der ID mit führender Null, Eingabefehler wird nicht korrigiert (Behinderung der Response Typ 2)
Quelle(n)	S3 – Usability-Problem bekannt aus FAQ
Modifikation(en)	S3 – Ergänzen von automatischer Behandlung des Fehlers (SA-Sensitivität Typ 1), einfachste Lösung ist Auslassen des Tests (keine SA-Sensitivität)
Gesamtbewertung	(2, 1)

Tabelle 45: Argumentationspfad des Interaktionsszenarios Nachsichtiges Format in Fallstudie TS

*P*rojekt	TS
Nutzungskontext	Aufgabe (Walkthrough)
Interaktionsklasse(n)	Erfassung und Behandlung von Benutzereingaben; Behandlung von Fehlern und Hilfe
Interaktionsszenario	*14. Prüfen auf Korrektheit*
Usability-Anforderung	Stimulus – „Das System möchte sicherstellen, dass die vom Nutzer eingegebenen Daten valide und korrekt sind.“, Response – „Sobald der Nutzer das Formular abschickt, überprüft das System die Eingaben auf Fehler. Wenn es Fehler gibt, startet das System eine Fehlerbehandlungsroutine.“
Use Case(s)	Login
Entscheidung(en)	S4 – Speicherung der Login-Daten (User-ID) als Integer (Behinderung der Response Typ 1)
Quelle(n)	S4 – von Quelltext generierter Dokumentation (Signaturen)
Modifikation(en)	S4 – Änderung einer Komponente (SA-Sensitivität Typ 1): Auslassen der Prüfung oder Kooperation mit Server
Gesamtbewertung	(1, 1)

Tabelle 46: Argumentationspfad des Interaktionsszenarios Prüfen auf Korrektheit in Fallstudie TS

Die Interaktionsszenarios 5 und 14 sind ähnlich, denn sie betrachten ein Problem aus zwei Perspektiven. Das Interaktionsszenario „Nachsichtiges Format“ betrifft die

Flexibilität der Anwendung *während* der Eingaben, das Interaktionsszenario „Prüfen auf Korrektheit“ den Test *nach* den Eingaben. Die Entscheidungen S3 und S4 betreffen demnach auch verschiedene Bearbeitungsschritte einer Interaktion. In der Diskussion in Phase 4 wurden die Entscheidungen deshalb in einem Thema zusammengefasst.

*P*rojekt	TS
Nutzungskontext	Hauptinteraktion
Interaktionsklasse(n)	Adaptation (durch Benutzer, für Benutzer, für Aufgaben); Ausführen, Wiederholen und Zurücknehmen von Befehlen
Interaktionsszenario	*49. Kontextbewusste Interaktion*
Usability-Anforderung	Stimulus – „Benutzer möchten eine kontextabhängige Aufgabe erfüllen.“, Response – "Das System gleicht die Auswahl der zur Verfügung gestellten Werkzeuge/Aktionen beständig mit dem sich verändernden Kontext ab.“
Use Case(s)	Staumeldung
Entscheidung(en)	S5 – System fordert Benutzereingabe basierend auf aktueller Geschwindigkeit (keine Behinderung der Response) S6 – Aktuelle Daten werden zur Staumeldung verwendet, statt die, die zur Staumeldung geführt haben (Behinderung der Response Typ 2, wenn Ortserkennung ausfällt)
Quelle(n)	S5 – Dokumentation S6 – Dokumentation
Modifikation(en)	S5 – keine Änderung notwendig S6 – Speicherung der Daten, die zur Stauerkennung geführt haben, notwendig ist Änderung mehrerer Komponenten (SA-Sensitivität Typ 1)
Gesamtbewertung	(2, 1)

Tabelle 47: Argumentationspfad des Interaktionsszenarios Kontextbewusste Interaktion in Fallstudie TS

Projekt	TS
Nutzungskontext	Umgebung, Mobile Anwendung
Interaktionsklasse(n)	Kooperation (in einem System, mit anderen Systemen)
Interaktionsszenario	*24. Konfliktfreie Nebenläufigkeit*
Usability-Anforderung	Stimulus – „Benutzer möchten mit mehreren Anwendungen gleichzeitig arbeiten.“, Response – „Das System kann gleichzeitig mit anderer Software betrieben werden, ohne dass es zu Konflikten wegen Nebenläufigkeit kommt.“
Use Case(s)	Parallelbetrieb Telefonieren
Entscheidung(en)	S7 – Parallelbetrieb ist nicht möglich (Behinderung der Response Typ 2)
Quelle(n)	S7 – Dokumentation über die Kommunikationskomponente, Befragung des Architekten
Modifikation(en)	S7 – Änderung der Communication-Komponente (SA-Sensitivität Typ 1), Neue Anforderung: Integration mit externer Hard- und Software von Navigationsgeräten
Gesamtbewertung	(2, 1)

Tabelle 48: Argumentationspfad des Interaktionsszenarios Konfliktfreie Nebenläufigkeit in Fallstudie TS

Projekt	TS
Nutzungskontext	Aufgabe (Auswahl aus Katalog)
Interaktionsklasse(n)	Behandlung von Fehlern und Hilfe, Kooperation (in einem System, mit anderen Systemen)
Interaktionsszenario	*25. Geräteunabhängigkeit*
Usability-Anforderung	Stimulus – „Benutzer möchten ein neues Endgerät installieren und verwenden.“, Response – „Selbst wenn Probleme durch auftretende Konflikte mit dem Gerät auftreten, können Benutzer das Gerät zum Interagieren mit der Anwendung verwenden.“
Use Case(s)	Integration mit externen Programmen bzw. Geräten (konkret: externen GPS-Empfänger verwenden)
Entscheidung(en)	S8 – Integration mit externem GPS-Empfänger (ok)
Quelle(n)	S8 – Dokumentation über die Kommunikationskomponente
Modifikation(en)	S8 – keine Änderungen notwendig
Gesamtbewertung	(0, 0)

Tabelle 49: Argumentationspfad des Interaktionsszenarios Geräteunabhängigkeit in Fallstudie TS

Projekt	TS
Nutzungskontext	Aufgabe (Katalog)
Interaktionsklasse(n)	Ausführen, Wiederholen und Zurücknehmen von Befehlen; Orientierung
Interaktionsszenario	*32. Fortschrittsanzeige*
Usability-Anforderung	Stimulus – „Benutzer möchten über den Fortschritt oder die verbleibende Zeit bis zum Ende eines Vorganges informiert werden“, Response – „Der Fortschritt des Vorgangs sowie die benötigte Zeit werden an ein Interface-Element fortwährend übermittelt und von diesem dargestellt.“
Use Case(s)	Anzeige des Sendens, Anzeige des Suchlaufs für externe GPS-Geräte
Entscheidung(en)	S9 – Nichtexistenz der Anzeige des Fortschritts (Behinderung der Response Typ 2)
Quelle(n)	S9 – aus Quelltext generierte Dokumentation
Modifikation(en)	S9 – Anzeigen des Status (SA-Sensitivität Typ 1), inkl. Status „sende gerade“ und dessen Anzeige im UI
Gesamtbewertung	(2, 1)

Tabelle 50: Argumentationspfad des Interaktionsszenarios Fortschrittsanzeige in Fallstudie TS

Projekt	TS
Nutzungskontext	Umgebung (Auswahl aus Katalog)
Interaktionsklasse(n)	Interagieren mit unbekannten Systemen; Strukturieren und Anzeigen von Content, Informationen, Daten; Navigieren, Browsen, Auswählen; Adaptation (durch Benutzer, für Benutzer, für Aufgaben)
Interaktionsszenario	*34. Vertrautes Aussehen und Verhalten*
Usability-Anforderung	Stimulus – „Benutzer möchten eine ihnen unbekannte Anwendung oder einen Teil davon nutzen.“, Response – „Das Aussehen und Verhalten der neuen Software entspricht der Software, die Nutzern schon bekannt ist.“
Use Case(s)	Navigieren
Entscheidung(en)	S10 – Unterstützung der plattformtypischen Softkeys, Anwendung der entsprechenden Styleguides (Response umgesetzt)
Quelle(n)	S10 – Dokumentation des User Interfaces
Modifikation(en)	S10 – keine Änderungen notwendig
Gesamtbewertung	(0, 0)

Tabelle 51: Argumentationspfad des Interaktionsszenarios Vertrautes Aussehen und Verhalten in Fallstudie TS

*P*rojekt	TS
Nutzungskontext	Mobile Anwendung (Auswahl aus Katalog)
Interaktionsklasse(n)	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Interaktionsszenario	*37. Prüfen notwendiger Ressourcen*
Usability-Anforderung	Stimulus – „Benutzer möchten einen Vorgang starten, der Ressourcen benötigt, die nicht immer vollständig verfügbar sind.“, Response – „Das System startet einen Vorgang nur dann, wenn es geprüft und bestätigt hat, dass alle erforderlichen Ressourcen verfügbar sind.“
Use Case(s)	Starten der Anwendung
Entscheidung(en)	S11 – Start der Anwendung nur, wenn genug Ressourcen vorhanden sind, Eigenschaft des Betriebssystems (Response umgesetzt)
Quelle(n)	S11 – Befragung des Architekten
Modifikation(en)	S11 – keine Änderungen notwendig
Gesamtbewertung	(0, 0)

Tabelle 52: Argumentationspfad des Interaktionsszenarios Prüfen notwendiger Ressourcen in Fallstudie TS

Projekt	TS
Nutzungskontext	Aufgabe (Auswahl aus Katalog)
Interaktionsklasse(n)	Behandlung von Fehlern und Hilfe; Strukturieren und Anzeigen von Content, Informationen, Daten
Interaktionsszenario	*43. Fehlermeldungen*
Usability-Anforderung	Stimulus – „Das System gibt eine Fehlermeldung aus.“, Response – „Nutzer erhalten kurze spezifische Informationen, um das Problem zu lösen. Die Fehler werden protokolliert und enthalten kodierte Informationen über die Fehlerkonditionen für die Entwickler.“
Use Case(s)	Rückmeldung bei Fehlern
Entscheidung(en)	S12 – Existenz von Fehlerdefinitionen und Exception Handling (ok) S13 – Existenz von Fehlermeldungen mit Informationen, was getan werden muss (ok) S14 – Satellitenpositionssystem liefert jede Sekunde Koordinate oder Fehlercode (ok) S15 – Communication-Komponente meldet Verfügbarkeit der Verbindungen (ok)
Quelle(n)	S12 – Quelltextdokumentation S13 – Dokumentation über die Benutzerschnittstelle S14 – Dokumenation der SA S15 – Dokumentation der SA
Modifikation(en)	S12 – keine Änderungen notwendig
Gesamtbewertung	(0, 0)

Tabelle 53: Argumentationspfad des Interaktionsszenarios Fehlermeldungen in Fallstudie TS

*P*rojekt	TS
Nutzungskontext	Aufgabe (Auswahl aus Katalog)
Interaktionsklasse(n)	Behandlung von Fehlern und Hilfe
Interaktionsszenario	*47. Mehrstufige Hilfe*
Usability-Anforderung	Stimulus – „Benutzer brauchen Hilfe im Umgang mit dem System.“, Response – „Das System stellt kontextabhängig die geeignete Art Hilfe zur Verfügung.“
Use Case(s)	Navigieren
Entscheidung(en)	S16 – Nichtexistenz einer spezialisierten Hilfskomponente (Behinderung der Response Typ 2)
Quelle(n)	S16 – UML-Diagramm, Befragung des Architekten
Modifikation(en)	S16 – Ergänzen einer Hilfskomponente und Verbindung mit anderen Komponenten (SA-Sensitivität Typ 2)
Gesamtbewertung	(2, 2)

Tabelle 54: Argumentationspfad des Interaktionsszenarios Mehrstufige Hilfe in Fallstudie TS

6.3.3 *Zusammenfassung der Ergebnisse*

In der Fallstudie SYM wurde keine der vier geprüften Interaktionsszenarios vollständig in der Architektur des Prototypen berücksichtigt (Tabelle 55). Ein Interaktionsszenario erfordert Änderungen der Kollaborationen der Elemente der Architektur, zwei Interaktionsszenarios erfordern strukturelle Änderungen der Architektur und der Modifikationsaufwand eines weiteren Interaktionsszenarios ist noch nicht bestimmbar.

Interaktionsszenario	*Projekt*	*Bewertung*
1. System-Feedback	SYM	(2, 2)
4. Abbrechen (Cancel)	SYM	(2, 1)
21. Add New Input/Output Device (Multi-Channel Access)	SYM	(2, 3)
26. Wiederherstellung nach Betriebsausfall (Recovering From Failure)	SYM	(2, 2)

Tabelle 55: Bewertung der Interaktionsszenarios in Fallstudien SYM, (Behinderung der Response, Einfluss auf Architektur)

Wie die Tabelle 56 im Detail auf Entscheidungsebene zeigt, erfordern von neun szenario-beeinflussenden Entscheidungen drei Entscheidungen keine Änderungen, zwei Entscheidungen erfordern Modifikationen der Kollaborationen der Elemente der bestehenden Softwarearchitektur auf der betrachteten Architekturebene, zwei Entscheidungen erfordern strukturelle Änderungen, zwei Entscheidungen erfordern Änderungen, deren Aufwand noch als unbekannt eingeschätzt wird.

Interaktionsszenario	*Anzahl S*	*Typ 0*	*Typ 1*	*Typ 2*	*Typ 3*
System-Feedback	3	1	1	1	0
Abbrechen	3	2	1	0	0
Multi-channel Access	2	0	0	0	2
Wiederherstellung nach Betriebsausfall	1	0	0	1	0
Insgesamt	9	3	2	2	2

Tabelle 56: Szenario-beeinflussende Entscheidungen (S) der Fallstudie Shake Your Mac (SYM) und ihre Klassifikation

Tabelle 57 fasst zusammen, dass in Fallstudie TS vier der zwölf geprüften Interaktionsszenarios vollständig in der Architektur des Prototypen berücksichtigt wurden. Von acht noch umzusetzenden Anforderungen erfordern fünf Interaktionsszenarios Änderungen der Kollaborationen der Elemente, drei Interaktionsszenarios erfordern strukturelle Änderungen der Architektur.

Interaktionsszenario	*Projekt*	*Bewertung*
1. System-Feedback	TS	(2, 2)
4. Abbrechen (Cancel)	TS	(2, 2)
5. Forgiving Format (Nachsichtiges Formular)	TS	(2, 1)
14. Checking for Correctness (Prüfen auf Korrektheit)	TS	(1, 1)
24. Using Applications Concurrently (Konfliktfreie Nebenläufigkeit, Non-conflicting Application Concurrency)	TS	(2, 1)
25. Maintaining Device Independence (Geräteunabhängigkeit, Device Independence)	TS	(0, 0)
32. Progress Indication (Fortschrittsanzeige)	TS	(2, 1)
34. Familiar Appearance and/or Behaviour (Vertrautes Aussehen und Verhalten)	TS	(0, 0)
37. Verifying Resources (Prüfen notwendiger Ressourcen)	TS	(0, 0)
43. Error Message (Fehlermeldungen)	TS	(0, 0)
47. Help (Mehrstufige Hilfe)	TS	(2, 2)
49. Workflow Model (Workflow-Modell, Context-aware Interaction)	TS	(2, 1)

Tabelle 57: Bewertung der Interaktionsszenarios in Fallstudie TS (Behinderung der Response, Einfluss auf Architektur)

Wie Tabelle 58 im Detail auf Entscheidungsebene zeigt, erfordern von 13 szenario-beeinflussenden Entscheidungen fünf Typ-0-Entscheidungen keine Änderungen, fünf Typ-1-Entscheidungen erfordern Modifikationen des Verhaltens der Softwarearchitektur, drei Typ-2-Entscheidungen erfordern strukturelle Änderungen.

Interaktionsszenario	*Anzahl S*	*Typ 0*	*Typ 1*	*Typ 2*	*Typ 3*
System-Feedback	1	0	0	1	0
Abbrechen	1	0	0	1	0
Nachsichtiges Format	1	0	1	0	0
Prüfen auf Korrektheit	1	0	1	0	0
Konfliktfreie Nebenläufigkeit	1	0	1	0	0
Geräteunabhängigkeit	1	1	0	0	0
Fortschrittsanzeige	1	0	1	0	0
Vertrautes Aussehen und Verhalten	1	1	0	0	0
Prüfen notwendiger Ressourcen	1	1	0	0	0
Fehlermeldungen	1	1	0	0	0
Mehrstufige Hilfe	1	0	0	1	0
Kontextbewusste Interaktion	2	1	1	0	0
Insgesamt	13	5	5	3	0

Tabelle 58: Szenario-beeinflussende Entscheidungen (S) der Fallstudie TrafficScanner (TS) und ihre Klassifikation

Zusammenfassend (Tabelle 59) wurden 22 szenario-beeinflussende Entscheidungen ermittelt. Acht erfordern keine Modifikationen, da sie die Response ihres Interaktionsszenarios erfüllen. Sieben erfordern einfache Modifikationen, fünf erfordern komplexe Modifikationen. Der Modifikationsaufwand von zwei Entscheidungen war zum Zeitpunkt der Analyse nicht bestimmbar. Die Softwarearchitektur der Anwendung TrafficScanner wurde mit rund 77% guten bzw. einfach zu ändernden szenario-beeinflussenden Entscheidungen und rund 23% komplexen Änderungen besser bewertet als die Softwarearchitektur der Anwendung Shake Your Mac mit rund 56% guten bzw. einfach zu ändernden szenario-beeinflussenden Entscheidungen und rund 44% komplexen oder unbekannten Änderungen.

Interaktionsszenario	*Anzahl S*	*Typ 0*	*Typ 1*	*Typ 2*	*Typ 3*
Insgesamt SYM	9	3 (33%)	2 (22%)	2 (22%)	2 (22%)
Insgesamt TS	13	5 (38%)	5 (38%)	3 (23%)	0 (0%)
Alle	22	8 (36%)	7 (32%)	5 (23%)	2 (9%)

Tabelle 59: Szenario-beeinflussende Entscheidungen (S) der Fallstudien SYM und TS und ihre Klassifikation

Die beiden folgenden Abbildungen 80 und 81 fassen diese detaillierten Informationen übersichtlich zusammen. Das Hauptziel Usability wurde durch die Rahmenbedingungen genauer spezifiziert, diesen wurden Interaktionsszenarios zugeordnet. Die davon abgeleiteten und analysierten Use Cases förderten Architekturentscheidungen mit Einfluss auf die Usability bzw. die Softwarearchitektur zu Tage.

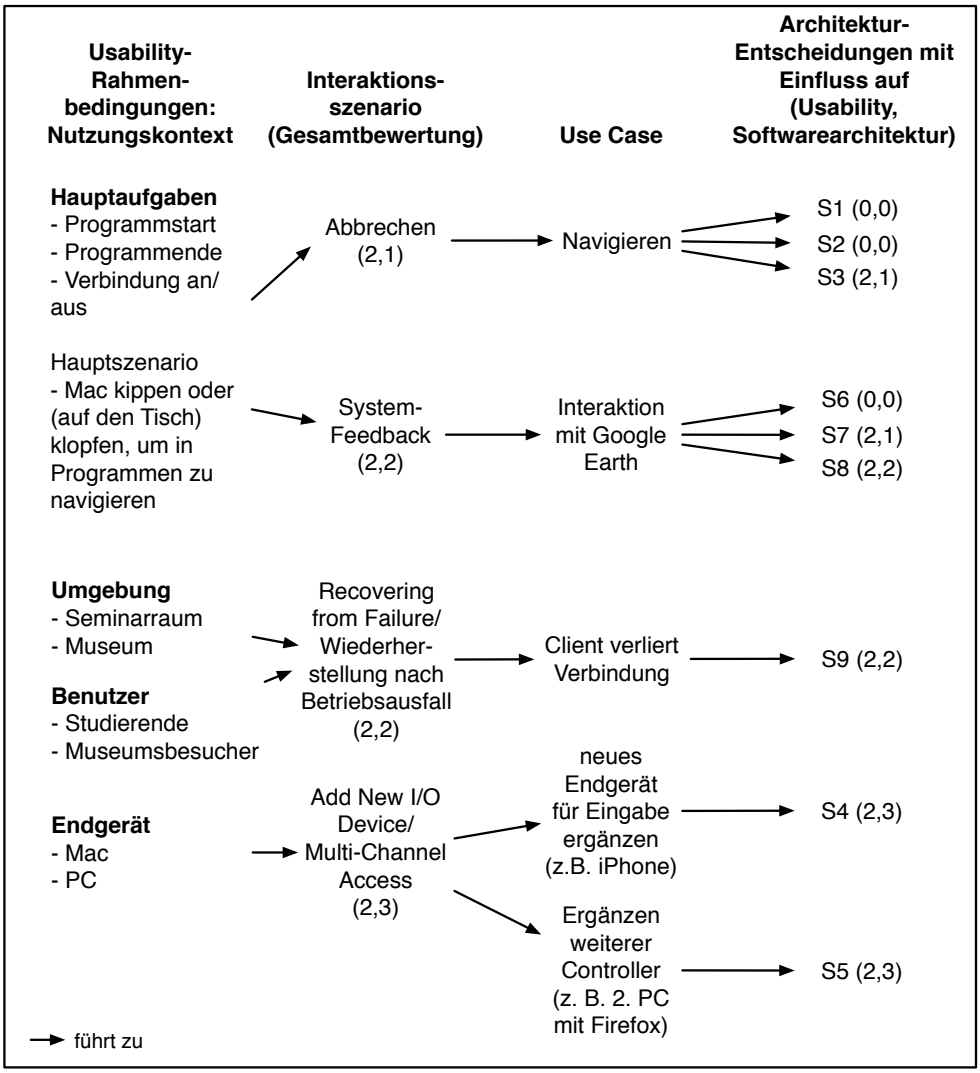

Abbildung 80: von Interaktionsszenarios über Use Cases zu Architekturentscheidungen (SYM)

In der ersten Fallstudie wurde ermittelt, dass die Benutzung der Anwendung voraussichtlich wegen Feedback-Problemen durch die verwendete Queue für die Ausgabeinterfaces und damit verbundene Verzögerungen beim Navigieren sehr schwierig ist; bestätigt wurde die Erkenntnis, da die Nutzer aufgrund des Antwortverhaltens Probleme während des Nutzertests hatten. Des Weiteren wurde ermittelt, dass ein automatischer Neustart der Verbindung nicht funktionieren kann und dass der Prototyp nicht leicht erweitert werden kann, es sei denn, Komponenten würden anders auf den Eingabe- und Ausgabegeräten verteilt werden; dies konnte nur theoretisch erläutert werden.

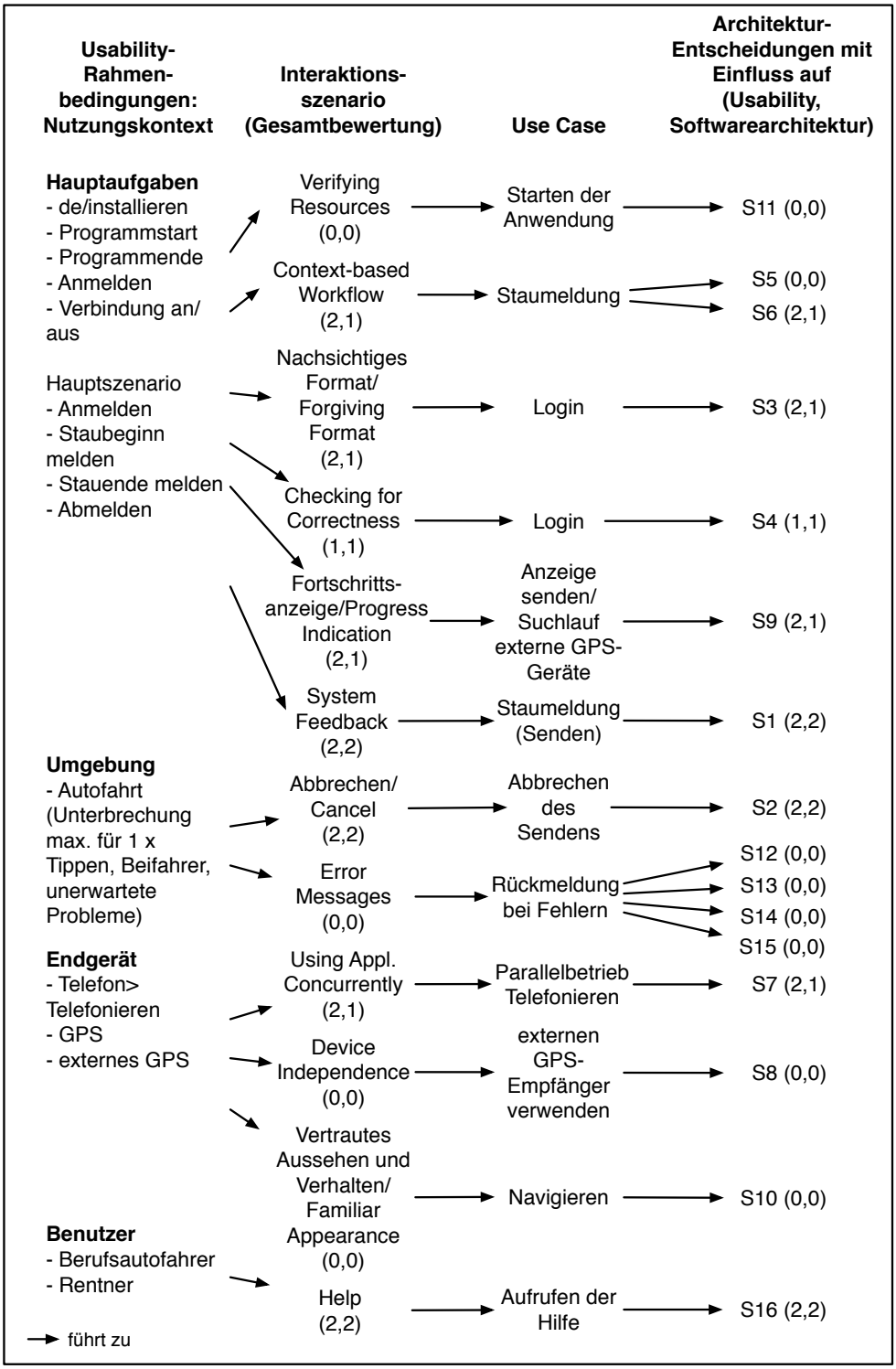

Abbildung 81: von Interaktionsszenarios über Use Cases zu Architekturentscheidungen (TS)

In der zweiten Fallstudie wurde eine praxiserprobte Anwendung untersucht, deren Softwarearchitektur weniger gravierende Usability-Probleme verursacht. Problematisch ist allerdings, dass wegen des Nichtspeicherns von Positionsdaten zum Meldezeitpunkt selbst kürzeste GPS-Verbindungsabbrüche dazu führen, dass die Anwendung die Interaktion zwar startet, dann aber abbricht; somit wird der Fahrer beim Autofahren unnötig vom Fahren abgelenkt und kann keine Meldung senden. Ermittelt

wurde außerdem, dass ein ursprünglich komplex erscheinender Fehler, der bei der Anmeldung auftrat und zu vielen Anfragen beim Service geführt hatte, einfach zu beheben ist, dass die Anwendung die grundlegende Funktion des Telefonierens behindert und dass weder das Abbrechen unterstützt wird noch eine Hilfe vorhanden ist.

Nachdem die abgeleiteten Ergebnisse genauer betrachtet wurden, bilden das Ergebnis verwischende Variablen einen weiteren Aspekt der internen Validität, der als nächstes behandelt wird.

6.3.4 *Das Ergebnis verwischende Variablen*

Es stellt sich die Frage, inwiefern der Erfolg der Methode von unterschiedlichen Vorkenntnissen abhängt. Eine gute Kenntnis der Unterlagen und der Interaktionsszenarios fördert die Effizienz der Durchführung der Methode SATURN, so dass die interne Validität durch die Vertrautheit mit einer Methode oder durchs Lernen unbeeinträchtigt bleibt. Sind die Unterlagen und die Wissensbasis nicht bekannt, ist die Analyse zeitaufwendiger. Der Einfluss des Vorwissens und der Erfahrung von Softwarearchitekten sollte in zukünftigen Forschungsprojekten (Forschungsprozessstufe 3) als Variable überprüft werden.

Argumentiert werden kann an dieser Stelle, dass SATURN im Vergleich zu den verwandten Arbeiten deutlich weniger Wissen voraussetzt:

- Die Teilnehmer werden durch eine Wissensbasis unterstützt. Die Wissensbasis setzt keine Kenntnisse im Bereich Usability oder Softwarearchitektur voraus, da sie aus Benutzersicht verfasst ist (Interaktionsszenarios, siehe Kapitel 3, S. 27 ff.).
- Die Analyse erfordert keine genaue Kenntnis der Pattern-Literatur, sondern orientiert sich an der Frage, ob die Anforderungen anhand der Architektur umgesetzt wurden. SATURN nutzt eine softwaretechnische (von Sichtenmodellen bekannte) Vorgehensweise bei der Erhebung von Entscheidungen (siehe Kapitel 4, S. 48 ff.). Im Gegenteil dazu erfordern die Methoden ATAM und SALUTA die Kenntnis von Patterns (siehe Kapitel 2, S. 8 ff.).

Beim Erarbeiten und Diskutieren von alternativen Lösungen spielt die Kreativität der Anwender dieser Methode eine Rolle. Wenn sie keine oder zu wenige Alternativen zu einer szenario-beeinflussenden Entscheidung finden und diskutieren können, so muss für die jeweilige Entscheidung ein unbekannter Änderungsaufwand angegeben werden, dann wird sie später im Anschluss an die SAA behandelt. In der vierten Phase von SATURN wird diese Situation in der Architektur-Support-Level (ASL)-Tabelle sichtbar. Diese drei Aspekte wurden als Wissensbasis und zur Bewertung der Architektur mit einbezogen.

6.3.5 *Zusammenfassung zur internen Validität*

Hinsichtlich der internen Validität wurde detailliert aufgezeigt, wie die Resultate nachvollziehbar von den Daten abgeleitet wurden. Mithilfe detaillierter Übersichten wurde gezeigt, dass anhand der Fallstudienergebnisse keine Invalidation stattfand. Verwischende Variablen wurden ermittelt und Maßnahmen, um ihren Einfluss zu verringern, wurden ergriffen. Die Methode SATURN ist also intern valide.

6.4 EXTERNE VALIDITÄT

Das Forschungsergebnis, also die Methode SATURN und ihre Erprobung, sollen zu einem Prozess der analytischen Verallgemeinerung beitragen.

Qualitative Studien unterstützen die empirische Induktion, d.h. die Beweiskraft ist darauf aufgebaut, dass nachfolgende Studien die Theorie ebenso unterstützen wie vorhergehende Studien. Diese analytische Generalisierung ist stärker als die statistische Generalisierung, da sie nicht auf statistische Korrelation verweisen muss, sondern zugrunde liegende Mechanismen genau untersucht und ein deutlich umfassenderes und realistischeres Abbild der Realität ermöglicht.[ESSD07]

Die Fallstudie SYM mit einer einfachen Anwendung und die nachfolgende Fallstudie TS mit einer komplexeren Anwendung, bestätigten die Machbarkeit von SATURN. Sowohl die in den Fallstudien behandelten Anwendungen als auch die Teilnehmer der Studien sind repräsentativ für ein reales Umfeld, in dem Softwarearchitekturanalysen stattfinden.

In beiden Fallstudien wurden mobile Anwendungen analysiert und der Einfluss von Architekturentscheidungen auf Interaktionsszenarios offengelegt. Die Fallstudien zeigten eine erfolgreiche Anwendung der Methode und bei der ersten Fallstudie, wie sie mit einer User Evaluation kombiniert werden kann. Somit wurde die Machbarkeit der Methode unter realistischen Bedingungen erprobt und bestätigt.

Der zweite Aspekt der externen Validität ist die Repräsentativität der Teilnehmer der Studien. Die erste Fallstudie wurde von einem Analysten und einem Architekten durchgeführt, die beide über Kenntnisse im Bereich Usability und Softwareentwicklung verfügen. Es ist möglich, dass diese Kombination nicht häufig existiert, aber unrealistisch ist diese Situation nicht. Üblich ist die Konstellation der Fallstudie 2, in der die SA-Analysemethode mit einem Architekten durchgeführt wurde, der Experte über die Softwareentwicklung mobiler Anwendungen ist.

Mit SATURN wurde ein Vorgehen entworfen, mit dem der Zusammenhang zwischen Usability und Softwarearchitektur im Detail analysiert werden kann.

Forschungsziel war die Erstellung und Erprobung einer solchen Methode. Um analytische Generalisierbarkeit zu erreichen, sind zahlreiche weitere Fallstudien im Anschluss an diese Forschung notwendig. Unter der Einschränkung der Forschungsziele und der Forschungsmethodik und der damit zusammenhängenden Anzahl und Ausprägung der Fallstudien, wird somit die Unterstützung der analytischen Generalisierung bestätigt.

Die Fallstudien SYM und TS bestätigen die Machbarkeit von SATURN. Die Repräsentativität der mobilen Anwendung und der Teilnehmer ist bei der ersten Fallstudie eingeschränkt, bei der zweiten Fallstudie gegeben. Mit dem Entwurf von SATURN in dieser Arbeit wurden weitere Forschungen hinsichtlich des architektonischen Einflusses von Usability vorbereitet.

6.5 NÜTZLICHKEIT

6.5.1 *Ziele von Softwarearchitekturanalysemethoden*

Wie schon in den Kapiteln 2 und 4 beschrieben, sollen Softwarearchitekturanalysemethoden ermitteln, ob eine Softwarearchitektur bestimmte Qualitätsmerkmale unterstützt und ob notwendige Änderungen besonders aufwendig sind [DN02]. Konkretere, veränderte [BCK03] und neue Anforderungen werden dabei erkannt, die nicht auf Kundenwünschen, sondern auf Erkenntnissen aus der Architekturanalyse basieren [HHP00]. Mit diesen Untersuchungen sollen die Wahrscheinlichkeit von Risiken verringert, Qualitätsanforderungen verifiziert und Abhängigkeiten zwischen Architekturentscheidungen verstanden werden [PS15]. Nicht zuletzt sollen aktuelle Systembeschreibungen erstellt werden, um ein System zu verstehen und zu verwalten [CKK02] und den Einfluss von Architekturentscheidungen [Kru04] auf Anforderungen zu erkennen [SAR02].

6.5.2 *Zielerreichungsgrad in Fallstudien*

Erstens ging es in der Fallstudie (Abschnitt 5.2, S. 86ff.) „Shake Your Mac“ (aus der Forschung) darum, den aktuellen Stand der Anwendung zu verstehen, denn es sollte eine neue Version des Prototypen entwickelt werden. Zweitens sollte geklärt werden, inwiefern Usability-Anforderungen unterstützt werden. Nach Aussagen des Architekten war die Analyse effizient und führte zu interessanten Erkenntnissen und neuen Anforderungen. Das erste Ziel wurde erreicht, indem die aktuelle Architektur analysiert wurde; für das zweite Ziel wurden die ausgewählten Interaktionsszenarios bewertet. Diskussionen gaben zudem wertvolle Impulse für die anstehende Arbeit an der Architektur. Insbesondere betrifft das neue Anforderungen, zu denen offengelegt wurde, wie die Architektur modifiziert werden soll.

Nützlich war außerdem, die Ergebnisse eines auf den gleichen Anforderungen basierenden Nutzertests mit den Ergebnissen der Architekturanalyse abzugleichen (Abschnitt 5.2.8, S.111): es wurden insgesamt mehr Usability-Probleme ermittelt; es ergaben sich Ursache-Wirkungs-Probleme durch das Mapping der Ergebnisse von Nutzertest und Architekturanalyse; der sehr hohe Schweregrad des Response-Problems in der Architekturanalyse wurde verifiziert; neue Interaktionsszenarios wurden für die Architekturanalyse ermittelt. Damit erübrigt sich eine erneute Auswahl von Interaktionsszenarios (Effizienzsteigerung bei der Architekturanalyse). Aus der höheren Anzahl und besseren Qualität der Untersuchungsergebnisse folgt, dass weniger Architektur- und User-Interface-Iterationen nötig sind.

Die zweite Fallstudie (Abschnitt 5.3, S. 123ff.) zur Anwendung „TrafficScanner“ (aus der Praxis) zielte auf ein Reengineering der Softwarearchitektur dieser mobilen Anwendung ab, um Probleme der alten Version in einer neuen Version zu vermeiden. Aus Sicht des Architekten wurde das Hauptziel erreicht: die Softwarearchitektur wurde skizziert und das Reengineering vorbereitet. Des Weiteren wurde die Analyse, die verwertbare Ergebnisse für die aktuelle Projektarbeit bereithält, als schnell und unkompliziert beschrieben. Außerdem sollen die Interaktionsszenarios für interne Tests verwendet werden.

Beide Fallstudien waren also aus Sicht der Personen, die sie durchgeführt haben, nützlich. Sie wurden außerdem beide veröffentlicht. Die wissenschaftliche Fallstudie mit der Kombination des Nutzertests und SATURN erschien im *Journal of Software and Systems (JSS)* [BGG10]. Die Industriefallstudie erschien in den Proceedings der *36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA)* [BG10a].

6.5.3 *Aufwand und Nutzen der betrachteten Methoden*

6.5.3.1 *Ressourcen- und Zeitaufwand*

Generell ist bei ATAM der Ressourcen- und Zeitaufwand am höchsten, da sich alle Stakeholder beteiligen. Selbst für kleine Untersuchungen werden 25 bis 50 Personentage veranschlagt [PS15]. Wegen des Formats der qualitätsattributsbasierten Usability-Szenarios soll für ATAM außerdem der „Usability-Quality-Attribute-Workshop“ [RRD07] besucht werden (siehe Abschnitt 3.2.2, S. 30).

Diese Art der Vorbereitung ist bei SALUTA und SATURN unnötig. Und da nur Architekten und Analysten teilnehmen (Stakeholder nur vereinzelt), ist der Ressourcen- und Zeitaufwand bei SALUTA und SATURN deutlich geringer.

ATAM und SALUTA sind durch nichtdefinierte Fragen und eine freie Analyse sehr stark von der Expertise zu Patterns und zur Softwarearchitektur abhängig (Abschnitt 2.6, ab S. 20). SATURN fokussiert sich mit vorgegebenen Fragen auf Anforderungen und stützt die Analyse auf Sichtenmodelle, d.h. weder Expertise zu Architektur noch Patterns

wird benötigt; nicht zuletzt ist die Bewertung geregelt (Abschnitt 4.2, ab S. 62). So ist die Analyse vereinfacht und weniger abhängig von Experten.

6.5.3.2 *Nutzen*

ATAM untersucht alle gewünschten Qualitätsmerkmale, die anderen beiden Methoden betrachten nur die Usability; dennoch liefern SALUTA und SATURN genauere Aussagen. ATAM kann Risiken aufdecken, SALUTA aber erkennt integrierte Usability-Properties und Usability-Patterns (Abschnitt 2.6, ab S. 20). SATURN folgt einer anderen Strategie und sucht nicht nach Patterns, sondern nach Architekturentscheidungen, die ein Interaktionsszenario theoretisch unterstützen, be- oder verhindern und andere Qualitätsmerkmale beeinflussen. Damit ist diese Methode davon unabhängig, ob Patternsammlungen komplett sind oder ob Muster, die in einer Architektur vorkommen, wirklich vorteilhaft sind. (Abschnitt 4.2, ab S. 62).

Die Aussagekraft der Methoden ist unterschiedlich: in SALUTA werden grobe Richtungsaussagen darüber getroffen, ob Usability unterstützt wird (Abschnitt 2.4.2, ab S.12). ATAM nennt Risiken und Risikothemen, d.h. betrachtet damit wenige problematische Architekturentscheidungen, die aber wegen des Vorgehensmodells nicht mehr zu ihren ursprünglichen Szenarios zurückführbar sind (Abschnitt 2.5.2, ab S.16). Bei SATURN wird, ähnlich wie bei SALUTA, die Bewertung jedes Interaktionsszenarios aufgeführt. SATURN beschreibt allerdings die architektonische Unterstützung von Usability, d.h. welche Auswirkungen sich für die Architektur, für die Usability und andere Qualitätsmerkmale voraussichtlich ergeben und welche Architekturänderungen möglich wären. Die Aussagekraft der Ergebnisse von SATURN ist am höchsten.

6.5.3.3 *Fazit*

Generell werden mit allen Methoden die anfangs genannten Ziele erreicht: sie verifizieren Usability in verschiedenen Abstraktionsstufen und prüfen, ob Änderungen notwendig sind. Sie fördern neue Anforderungen zu Tage und ermöglichen es, ein System zu verstehen und zu verwalten.

Zusammengefasst ist der Aufwand für die Methode ATAM wegen des Personalaufwands, der Vorbereitung und der Fragetechnik am höchsten; gefolgt von der Methode SALUTA, die weniger Personalaufwand erfordert, aber die Kenntnis der Usability-Propertys und Usability-Patterns voraussetzt und jedes Szenario mit diesen ohne konkretes Vorgehen abgleicht. Die Methode SATURN ist vorm Hintergrund des Personalaufwandes, der vorbereiteten Interaktionsszenarios und der konkreten Fragetechnik am wenigsten aufwendig. Sie ist gleichzeitig die Methode mit der größten Aussagekraft. SATURN ist also, verglichen mit den früheren Methoden, am nützlichsten.

Methode	*Bewertung des Aufwands*	*Bewertung des Nutzens*
ATAM	[-] vorbereitender Workshop [- -] Workshops mit allen Stakeholdern [- -] Expertise zu unbestimmter Anzahl von Patterns [- -] starke Abhängigkeit von Expertenwissen	[++] Vielzahl von Qualitätsmerkmalen untersucht [+] Risiken und Wechselwirkungen erkannt [-] Risiken und Wechselwirkungen schwierig zu Zielen zurückverfolgbar [-] Geringe Detailtiefe
SALUTA	[+] kein vorbereitender Workshop [+] Workshop mit Architekten [+] Verständnis von bestimmten Usability-Properties und Usability-Patterns [-] starke Abhängigkeit von Expertenwissen	[+] Usability untersucht [+] Aussage, ob Usability-Properties und Usability-Patterns unterstützt werden [+] Richtungsaussage, ob Usability unterstützt wird
SATURN	[+] kein vorbereitender Workshop [+] Workshop mit Architekt und Analyst [++] Vereinfachung/Konkretisierung: Sichtenmodelle statt Patterns [++] Vereinfachung: Fragenkatalog und Benotung der Anforderungskonformität	[+] Usability untersucht [++] bewertete Aussage, wie Usability unterstützt wird [++] konkrete Auswirkungen auf Architektur, Usability [+] Wechselwirkungen zu Qualitätsmerkmalen [+] mögliche Architekturänderungen
Fazit	SATURN und SALUTA am wenigsten Aufwand für Zeit und Ressourcen, am meisten ATAM. SATURN benötigt das geringste Vorwissen, gefolgt von SALUTA und ATAM.	Bezüglich Usability hat ATAM die geringste, SALUTA eine bessere, SATURN die größte Aussagekraft.

Tabelle 60: Aufwand und Nutzen

Tabelle 60 fasst zusammen, wie die Methoden vorgehen und zeigt, dass die Methode SATURN durch ihr vereinfachtes Vorgehen den geringsten Aufwand und durch die konkretesten Aussagen den größten Nutzen bietet.

6.5.4 *Allgemeine Kritik an szenario-basierten Methoden*

Patidar und Suman [PS15] nennen sieben Probleme der von ihnen untersuchten szenario-basierten Softwarearchitekturanalysen, die ihren Nutzen einschränken: nicht nachprüfbare Szenario-Abdeckung, der Ressourcen- und Zeitaufwand, wenig Schulungsunterlagen, die übliche Validation mit Fallstudien, wenig Werkzeugunterstützung, starke Abhängigkeit von Experten, fehlendes gemeinsames Verständnis von Spezifikationen bei Softwarearchitekten. Dieser Abschnitt beschäftigt sich damit, inwiefern SATURN diesen Problemen entgegengewirkt.

6.5.4.1 *Szenario-Abdeckung*

Die Szenario-Abdeckung (im Sinne von Test-Abdeckung) ist bei szenario-basierten Methoden schwierig zu untersuchen; um dagegen zu steuern, kann beispielsweise ein Framework versteckte Annahmen aufdecken [LRV99] oder eine Bewertungsmatrix Szenarios basierend auf Stakeholder-Zielen priorisieren [LBKK97]. [PS15]

In SATURN wird mit einer Nutzungskontextanalyse darauf hingearbeitet, dass die zweckmäßigste Nutzer-System-Interaktion und das Alleinstellungsmerkmal erkannt und in Interaktionsszenarios überführt wird (Abschnitt 4.2.3, ab S. 68). Regeln zur Auswahl und eine offene und dokumentierte Diskussion unter den beteiligten Personen soll die Selektion von aus Nutzersicht relevanten Interaktionsszenarios ermöglichen (Abschnitt 4.2.4, S. 68). Eine Garantie, dass die perfekten Interaktionsszenarios ausgewählt oder beschrieben werden, kann es allerdings nicht geben. Durch die häufige Wiederholung und damit zusammenhängende kontinuierliche Verbesserung der Methode (Abschnitt 4.2.7, S. 73) und die (optionale) Kombination mit einem Nutzertests werden aber die Chancen verbessert, weitere Interaktionsszenarios für die Analyse zu ermitteln (z.B. wie bei Fallstudie SYM, Abschnitt 5.2.8.3, ab S. 114).

Eine bessere Szenario-Abdeckung kann auch durch einen Methodenmix erreicht werden. Ein Nutzertest und eine Architekturanalyse betrachten jeweils andere Aspekte einer Anwendung: Ein Nutzertest deckt Usability-Probleme beim Interagieren mit dem User Interface auf, z. B. weitere Themen in User Interface Design, Informationsarchitektur, Interaktionsdesign, die nur bei der Benutzung der Anwendung erkannt werden. Eine Architekturanalyse ermittelt die architektonische Unterstützung von Nutzer-System-Interaktionen. Gemeinsam können Nutzertest und Architekturanalyse eine größere Testabdeckung erreichen. Es ist also sinnvoll, sowohl Architekturanalyse als auch Nutzertest durchzuführen und miteinander zu kombinieren, um am Ende ein Produkt mit bestmöglicher Usability zu erhalten. (Abschnitt 4.3, ab S. 74)

6.5.4.2 *Ressourcen- und Zeitaufwand*

Patidar und Suman [PS15] sehen den Ressourcen- und Zeitaufwand für Untersuchungen bei ATAM sehr kritisch, sie sehen domänenspezifische Analysen als Lösung. SATURN fokussiert sich auf die Usability mobiler Anwendungen, ist also domänenspezifisch. Zudem erfordert diese Methode nur zwei Teilnehmer, die nur fünf Aktivitäten abarbeiten müssen (ATAM: alle Stakeholder, neun Aktivitäten in zwei Phasen). Weitere Argumente enthält Abschnitt 6.5.3.1, ab S. 192.

Für die Fallstudie SYM wurden zwei Personentage benötigt (Abschnitt 5.2.10.1, S. 121), für die Fallstudie TS vier Personentage (Abschnitt 5.3.9.1, S. 164). Es ist der zukünftigen Forschung vorbehalten, auch den personellen und zeitlichen Aufwand einzelner Phasen zu berücksichtigen.

6.5.4.3 *Schulungsunterlagen*

Benötigte Schulungsunterlagen sind eine weitere Herausforderung, wobei gerade reifere Methoden wie ATAM in Büchern gut und ausführlich beschrieben sind [PS15]. Diese Forschungsarbeit bietet sehr genaue Auskunft über die Methode und ihre Werkzeuge, inklusive der Interaktionsszenarios online (http://cassini.saturn-lab.com/) und im Anhang (Abschnitt A.2, ab S. 204). Erwähnt werden sollte an dieser Stelle auch, dass das ein Feedback zur zweiten Fallstudie in der Praxis war, dass die Analyse schnell und unkompliziert verlaufen ist (Abschnitt 5.3.7, S. 162). Generell sollten spätere Studien weitere Schulungsunterlagen thematisieren.

6.5.4.4 *Validation mit Fallstudien*

Die Methoden werden üblicherweise mit Fallstudien validiert. Praktiker sind so allerdings weniger von der Nützlichkeit der Methoden zu überzeugen, weshalb weitere Fallstudien im Wirtschaftsumfeld nötig sind. [PS15].

SATURN wurde ebenfalls mit Fallstudien validiert. Ziele war es hier, dass die theoretische Konstruktion über Literaturstudien, das Method Engineering und die praktische Erprobung konstruktiv, intern und extern valide sind. Eine Verallgemeinerung der Ergebnisse ist ein langfristiger Reifeprozess und liegt außerhalb dieser Forschungsarbeit (siehe Prinzip der empirischen Induktion, Abschnitt 6.4, ab S. 190).

Langfristig sollen SATURN und die Interaktionsszenarios immer wieder geprüft werden: dazu dient die Phase 5 „Retrospektive“ (Abschnitt 4.2.7, S. 73). Außerdem unterliegen die Interaktionsszenarios einem qualitätssichernden Interaktionszenario-Lebenszyklus (Abbildung 11, S. 45). Schließlich wird mit dem Canonical Action Research rigoros daran gearbeitet, die Methode und ihre Hilfsmittel zu optimieren (Abschnitt 5.1.3, ab S. 86). Somit fokussieren Methode, Hilfsmittel und Studiendesign einen kontinuierlichen und langfristigen Reifeprozess.

6.5.4.5 *Werkzeugunterstützung*

Da die szenario-basierte Softwarearchitekturanalyse meist informell und manuell erfolgt, gibt es wenig Werkzeugunterstützung, allerdings gibt es generische und die Evaluation nicht komplett unterstützende Werkzeuge wie das Architecture Evaluation Tool AET, ArchE design assistant, ArcheOpterix, Acme Simulator und DeSi [AZR11]. [PS15]

Auch SATURN muss aktuell manuell durchgeführt werden. Die Hilfsmittel sind, wie schon im Abschnitt 6.5.4.3 (S. 195) erwähnt, digital verfügbar. Das Kapitel 4 kann als Anforderungsbeschreibung für ein neu zu erstellendes Werkzeug verstanden werden (Kapitel 4, ab S. 48). Hier sollte besonders darauf geachtet werden, dass Analysen wiederholt durchgeführt werden können und dass Ergebnisse der verschiedenen Analysedurchgänge referenziert und verknüpft sein sollen, aber inhaltlich nicht rückwirkend geändert werden dürfen.

6.5.4.6 *Abhängigkeit von Experten*

Die von [PS15] untersuchten Methoden hängen stark vom Expertenwissen der Personen ab, die sie durchführen. Grundsätzlich gilt das für alle Methoden mit Fragetechniken: sie hängen von den Personen ab, die sie durchführen; sie sind als qualitative Untersuchungen konzipiert, mit allen dazu gehörenden Vor- und Nachteilen (Abschnitt 5.1.3, ab S. 86).

Generell ist deshalb ein Methodenmix üblich und sinnvoll, so dass qualitative und quantitative Methoden miteinander verknüpft werden. Wie das mit einem Nutzertest möglich ist, wurde im Rahmen dieser Arbeit untersucht (Abschnitt 4.3, ab S. 74).

In Kapitel 2 wurde beschrieben, dass die Pattern-Orientierung der früheren Methoden zu einer sehr großen Abhängigkeit von Expertenwissen zu Softwarearchitektur und Softwarearchitektur-Patterns führt. Das heißt, basierend auf ihrem Erfahrungsschatz und dokumentierten Patterns, stellen bei ATAM und SALUTA die Experten Fragen zur Architektur, die sie dann miteinander diskutieren. Aus diesem Grund ist auch so schwierig nachvollziehbar, wie die eigentliche Analyse der Architektur erfolgt. (Kapitel 2, ab S. 8)

Die Forschungsfragen dieser Arbeit (Abschnitt 2.7, ab S. 25) beschäftigen sich insgesamt mit dem Thema Vereinfachung, insbesondere die Fragen 2, 3, 4, 5 zu den Hilfsmitteln und die Frage 6 zu einer Fragetechnik mit geringeren Freiheitsgraden. Erarbeitet wurden nun Interaktionsszenarios, mit denen es nicht mehr notwendig ist, Patterns zu

kennen, die leicht angepasst werden können oder als Vorlage dienen, die ein überprüfbares Soll-Verhalten der Anwendung definieren und von Patterns abgeleitet sind, die sie referenzieren. Freiheitsgrade der Fragetechnik wurden wie folgt eingeschränkt: vier standardisierte Fragen sind immer wieder zu stellen, um mit Interaktionsszenarios im Sinne der Sichtenmodelle (Abschnitt 4.2.1.4, ab S. 64) und der Architekturdefinition von Kruchten (Abschnitt 1.1, S. 1) die Struktur und das Verhalten von Komponenten und Schnittstellen zu erheben. (Abschnitt 4.4, ab S. 76)

Zukünftige Untersuchungen sollen sich weiterhin darauf fokussieren, Freiheitsgrade so gering wie möglich zu gestalten.

6.5.4.7 *Verständnis der Spezifikationen bei Softwarearchitekten*

Patidar und Suman [PS15] bemängeln, dass Softwarearchitekturdokumente oft nicht gut gepflegt und formalisiert werden, so dass oft ein gemeinsames Verständnis von Spezifikationen bei Softwarearchitekten fehlt.

Das wäre ein kritisches Problem, wenn sich eine Architekturanalyse auf vorhandene schriftliche Unterlagen beschränken würde. Bei SATURN gilt deshalb die Vorbedingung, eine Architekturbeschreibung vorzulegen (siehe Vorbedingung, Abschnitt 4.2.3, ab S. 68); insbesondere wird aber der aktuelle Status der Architektur in Phase 3 „Evaluieren der Interaktionsszenarios“ mündlich erfragt und schriftlich dokumentiert (Abschnitt 4.2.5, ab S. 69).

6.5.4.8 *Softwarequalität*

Das Prinzip der Anforderungsorientierung von SATURN könnte so strikt ausgelegt werden, dass auch schlechte Softwarequalität akzeptabel ist, wenn Interaktionsszenarios hypothetisch durchführbar sind. Das ist vor dem Hintergrund der Wechselwirkung mit anderen Zielen, z.B. gute Wartbarkeit, nicht sinnvoll.

Ein Interaktionsszenario besteht nicht nur aus einer beschriebenen Nutzer-System-Interaktion, sondern gibt auch Qualitätskriterien vor. Hier können Architekten entsprechende Qualitätsmerkmale mit in die Response-Prüfung aufnehmen und die Architektur später auch dahingehend bewerten (siehe Kernelemente eines Interaktionsszenarios in Tabelle 4, S. 32).

6.5.4.9 *Fazit*

Die allgemeinen Kritikpunkte werden von der Methode SATURN nachvollziehbar so behandelt, dass bekannte negative Einflüsse verringert werden. Mit weiteren Fallstudien und durch eine Werkzeugunterstützung der Analyse selbst, sollen die Methode und ihre Hilfsmittel auch im Sinne der empirischen Induktion kontinuierlich weiter verbessert werden.

6.6 ZUSAMMENFASSUNG

Im Detail wurden die Validationsziele wie folgt erreicht:

KONSTRUKTIVE VALIDITÄT

- ✓ Die Definitionen der einzelnen theoretischen Elemente sind wissenschaftlich korrekt und damit valide.
- ✓ Es wurde gezeigt, dass die Zusammenhänge zwischen ihnen logisch nachvollziehbar korrekt und damit valide sind.
- ✓ Es wurde gezeigt, dass die Vorgaben für die Bewertung wissenschaftlich korrekt und damit valide sind.

- ✓ Es wurde gezeigt, dass die Wahl der Skala (Ordinalskala) logisch nachvollziehbar korrekt und valide ist und dass keine unzulässige Umwandlung geschieht.

INTERNE VALIDITÄT

- ✓ Es wurde detailliert gezeigt, dass die Resultate theoretisch nachvollziehbar von den Daten abgeleitet werden.
- ✓ Anhand der Ergebnisse der Fallstudie fand keine Invalidation statt.
- ✓ Die das Ergebnis verwischenden Variablen wurden ermittelt. Maßnahmen zur Verringerung des Einflusses wurden ergriffen.

EXTERNE VALIDITÄT

- ✓ Die Resultate tragen zu einem Prozess der analytischen Verallgemeinerung bei.

Die Forschungsarbeit erreicht die Validationsziele der Stufe 1 und Stufe 2 und erfüllt Ziel und Anspruch des Forschungsprojektes. Die Methode SATURN ist eine valide Analysemethode, mit der von Usability-Anforderungen ausgehend architektursensitive Entscheidungen ermitteln werden können.

Die Untersuchung der Nützlichkeit plausibilisierte und argumentierte, dass SATURN aus drei Gründen nützlich ist. Erstens waren die Fallstudien aus Sicht der Personen, die sie durchgeführt haben, sinnvoll. Zweitens erreicht diese Methode die an szenariobasierte Methoden gestellten Ziele mit weniger Aufwand und mehr Nutzen als die betrachteten früheren Arbeiten. Drittens bietet sie nachvollziehbar sinnvolle Antworten auf allgemeine Probleme von szenario-basierten Softwarearchitekturanalysemethoden an.

Die selbst gesetzten Ziele dieses Forschungsprojektes wurden damit erreicht. Zukünftige Forschung soll auf empirische Reliabilität ausgerichtet werden.

Nachdem die Validität der Methode bestätigt wurde, folgt nun das abschließende Kapitel Zusammenfassung: Es thematisiert kurz die Beiträge der Arbeit und weitere Forschungsthemen.

7 FAZIT

7.1 WESENTLICHE BEITRÄGE DER ARBEIT

Mit der Methode SATURN können Architekten und Analysten die architektonische Unterstützung von Usability analysieren und bewerten. Die Methode und ihre Interaktionsszenarios (Kapitel 3, S. 3ff., Kapitel 4, S. 48ff. und Anhang A, S. 203ff.) wurden mit einem partizipativen Forschungsprozess – der kanonischen anwendungsnahen Forschung – in zwei Fallstudien erprobt (Kapitel 5) und als machbar, gültig und nützlich bestätigt (Kapitel 6).

7.1.1 *Architekturanalyse im Anforderungskontext*

Die vorgestellten früheren Methoden fokussieren Attribute von Qualitätsattributen. Da Usability-Attribute die Benutzung einer Anwendung beschreiben, ist diese Perspektive für dieses Qualitätsmerkmal ungeeignet. Die Methode SATURN wurde deshalb in den Anforderungskontext gestellt.

So werden in Phase 1 basierend auf dem Nutzungskontext Anforderungen ermittelt, in Phase 2 Interaktionsszenarios als Usability-Anforderungen spezifiziert, in Phase 3 bewertet, ob die Architektur diesen Anforderungen gerecht wird; und abschließend werden neue Anforderungen und offene Fragen in Phase 4 ermittelt. Die Betrachtung des Nutzungskontexts führt zu einer Liste benutzer- und systeminitiierter Interaktionen. Einige davon werden in Interaktionsszenarios formuliert. Die zentralen Interaktionen einer Anwendung werden durch den Nutzungs- und Geschäftskontext beschrieben und sind Muss-Anforderungen. Diese Informationen bilden die Grundlage dafür, welche Interaktionsszenarios ausgewählt werden. Usability wird in SATURN als eine Qualitätsanforderung verstanden, die durch konkrete benutzer- oder systeminitiierte Interaktionen spezifiziert wird, denen eine oder mehrere qualitative Anforderungen sowie Rahmenbedingungen zugeordnet sind, die auf dem Nutzungskontext und dem Geschäftskontext basieren. Zu diesen qualitativen Anforderungen gehören auch Austauschbeziehungen mit anderen Qualitätsmerkmalen, wie Sicherheit, Modifizierbarkeit, Performanz. Ein Interaktionsszenario, das nicht unterstützt wird, wird als eine nicht umgesetzte Usability-Anforderung verstanden, die zu einem Usability-Problem führen kann.

Schließlich stellen die Ergebnisse dar, inwiefern die betrachteten Interaktionsszenarios unterstützt werden; es werden zudem neue Anforderungen und offene Fragen erstellt. Nachdem die Anwender dieser Methode die Ergebnisse diskutiert haben, priorisieren sie die Interaktionsszenarios und damit auch die durch sie repräsentierten Usability-Anforderungen.

Verglichen mit den früheren Arbeiten ist dieser Analysekontext gut nachvollziehbar und rückvollziehbar. Der starke Fokus auf den Geschäfts- und Nutzungskontext unterstützt methodisch, dass die Interaktionen untersucht werden, die aus wirtschaftlicher Sicht relevant sind. Bestätigte Lösungen und neue oder geänderte Anforderungen sind nützliche Eingaben für den Prozess Entwurf der Architektur.

7.1.2 *Koordination mit dem Usability Engineering*

Frühere Methoden integrieren das Usability Engineering nur über einen Informationsfluss zur Softwarearchitekturanalysemethode. Allerdings ist das Fachgebiet Mensch-

Maschine-Interaktion interdisziplinär, also ist eine gute Koordination sinnvoll. Zudem kann eine SAA Usability-Probleme aufdecken, deren Ursachen in technischen Grenzen liegt, sie kann aber nicht abschließend bewerten, wie gut die Benutzbarkeit des Systems ist. Außerdem fehlen dem Usability-Engineering die technischen Kenntnisse, um die Ursachen von Usability-Problemen zu erkennen. Eine Kooperation ist deshalb notwendig.

Die Methode SATURN wurde auf eine interdisziplinäre Koordination mit dem Usability Engineering abgestimmt. Das betrifft Begriffe, Eingaben und Ausgaben sowie angewendete Techniken. Usability wird als ein messbares Teilziel der User Experience [PRS07] verstanden. Der Nutzungskontext und Usability-Anforderungen sind die Eingaben für Usability-Evaluationen und SATURN. Analog zum Vorgehen im Usability Engineering wird erst der Nutzungskontext betrachtet und in Usability-Anforderungen überführt; analog zum Vorgehen im Fachgebiet MMI [FJM05, Deu11]. Interaktionsszenarios beschreiben einzelne Interaktionen und sind damit auch in Usability-Evaluationen als Teil eines Testfalls verwendbar. Schließlich entspricht der Ergebnistyp Usability-Problem dem Ergebnistyp von Usability-Evaluationen; die Ergebnisse sind also vergleichbar. Ein Vorgehen dazu, wie eine SAA-Methode mit einer User-Evaluation kombiniert werden kann, wurde beschrieben und explorativ erprobt [BGG10].

Aus dieser Perspektive betrachtet, ist SATURN eine qualitative Methode, die mit einer quantitativen User-Evaluation sinnvoll kombiniert werden kann. Identifizierte Probleme und Fragen, die auf diesen Resultaten basieren, können in beide beteiligte Prozesse zurückfließen, also den Entwurf der SA und das Usability Engineering. Die Anzahl der gefundenen Usability-Probleme wurde erhöht, denn es gibt Probleme, die nur mit der einen oder der anderen Methode aufgedeckt werden können. Nicht zuletzt wurden bessere Erkenntnisse über Auswirkungen und Ursachen von Usability-Problemen ermittelt.

7.1.3 *Interaktionsszenarios*

Hilfsmittel der betrachteten früheren SAA werden bei der Evaluation von Szenarios eingesetzt. Dabei wird vorausgesetzt, dass die Anwender der jeweiligen Methode eine Vielzahl schon existierender Lösungen kennen oder über ein sehr umfangreiches Fachwissen verfügen. Es ist notwendig, dass zumindest alle Inhalte der Wissensbasis (insbesondere Patterns) bekannt sind. Allerdings erstrecken sich Architekturpatterns meist über mehrere Seiten kompakt dargestelltes Wissen. Zudem führt Innovation immer wieder zu neuen Patterns. Vollständigkeit kann also nicht gewährleistet werden. Es ist daher sinnvoll, Lösungsräume einzugrenzen.

Die Wissensbasis von SATURN besteht aus Interaktionsszenarios. Diese abstrahieren und referenzieren Patterns, die abstrakte Lösungen für ein Interaktionsszenario beschreiben. Gültige Interaktionsszenarios sind von Patterns abgeleitet, Patterns verschiedener Fachbereiche. Diese Wissensbasis integriert und referenziert unter anderem auch die Erkenntnisse der betrachteten früheren Methoden. Sie sind auch Beispiele für neu zu erstellende Interaktionsszenarios.

Diese Wissensbasis wird bereits bei der Bestimmung und Auswahl von Interaktionsszenarios eingesetzt; danach auch bei der Evaluation der Interaktionsszenarios. Das Vorgehensmodell von SATURN fördert neue Interaktionsszenarios, indem die Anwender der Methode Interaktionen beschreiben, bevor sie den Katalog der Interaktionsszenarios durchsehen. Ein definierter Lebenszyklus sichert ab, dass die Nutzer der Interaktionsszenarios diese pflegen und bewerten, inwiefern diese potenziell architektursensitive Usability-Anforderungen enthalten. Pflege und Konsolidierung dieses wiederverwendbaren Wissens wurde in das Vorgehensmodell integriert.

Diese Wissensbasis und ihre methodische Integration ermöglichen es, Interaktionsszenarios auszuwählen, die potenziell architektursensitiv sind und den Lösungsraum einzugrenzen. In diesem werden abstrakte Lösungen beschrieben, die sie auf ihre spezielle Problemstellung anwenden können. Es ist möglich, von Patterns verschiedener Fachbereiche zu abstrahieren und neue Patterns hinzuzufügen. Die langfristig ausgelegte Pflege der Wissensbasis baut nach und nach eine solide Datenbasis für die weitere Erforschung des Zusammenhangs von Softwarearchitektur und Usability auf.

7.2 WEITERFÜHRENDE FORSCHUNG

Aufbauend auf den Forschungsergebnissen dieser Arbeit sind weitere Themen interessant. Einige Forschungsgebiete sind die quantitative Forschung, die sich direkt anschließen kann; die Einbeziehung anderer Qualitätsmerkmale und die Erweiterung der konstruktiven Aspekte der Methode SATURN.

- Quantitative Forschung: Mit SATURN können viele Fallstudien erstellt werden, um den Zusammenhang zwischen Usability und Softwarearchitektur zu erforschen. Wie können diese Forschungsergebnisse allgemeingültig beschrieben werden (Wissensrepräsentation)? Wie können diese werkzeugunterstützt erhoben werden? Welche Zusammenhänge zwischen Usability und Softwarearchitektur können werkzeugunterstützt ermittelt werden?
- Einbeziehung von anderen Qualitätsmerkmalen: Da ein Zusammenhang von Usability und anderen Qualitätsmerkmalen besteht, sollen andere Analysemethoden mit SATURN verknüpft werden. Zur Analyse von Qualitätsmerkmalen bestimmter Fachgebiete ist es sinnvoll, Methoden aus den Fachgebieten anzuwenden. Wie kann die Methode mit dieser Komplexität umgehen (z. B. Anpassung der Kontextdefinition, Recherche neuer Interaktionsszenarios)? Wie können Aufwand und Nutzen einer komplexen Analyse ausbalanciert werden?
- Konstruktion: Wie können die konstruktiven Aspekte dieser Analysemethode ausgebaut werden? Ergibt sich damit ein neues und nützliches Vorgehen zur Konstruktion von Software vor dem Hintergrund der Anforderungen agiler Prozesse?

Teil I

APPENDIX

A ANHANG

A.1 VORLAGE FÜR EIN INTERAKTIONSSZENARIO

Name	*Hinweise zur Beschreibung*
Nummer	Anzahl bisheriger Szenarios plus eins
Kurzname	Interaktion als Wort oder Wortgruppe
Quelle	Verursacher des Stimulus (z. B. Persona, Benutzer, System, Teil des Systems, bestimmte Benutzertypen)
Stimulus	Interaktion mit dem Artefakt
Artefakt	System oder Teilsystem, bei dem der Stimulus ankommt und bei dem er verarbeitet wird (z. B. Benutzerschnittstelle, Plattform, Mobiles Endgerät, anderes System)
Umgebung	in welcher Situation, zu welchem Zeitpunkt die Interaktion stattfindet (z. B. zur Laufzeit) und/oder Umgebungstypen (z. B. Museum, öffentliche Verkehrsmittel, Büro, Seminarraum)
Response	Erwartete Reaktion des Artefakts auf den Stimulus
Response-Prüfung	Qualitätskriterien für die Response (z. B. bestimmte Responsivität der Anwendung, Latenzzeit, maximale Fehlerrate bei der Benutzung)
Usability-Pattern(s)	Pattern(s) referenzieren, von denen das Interaktionsszenario abgeleitet wurde (bzw. werden kann)
Interaktionskategorie(n)	Eine oder mehrere Kategorien abhängig von Stimulus und Response (Liste siehe Abschnitt 3.3.2, S. 33)
Usability-Ziel(e)	Ein oder mehrere Usability-Ziele und Begründung (positiv und negativ)
Potenzielle Architektursensitivität	Anhand der Quellen und Einschätzung durch Experten: Komplexität von architektonischen Änderungen bei Nachimplementierung (siehe Tabelle 21, S. 70) Angabe des Vertrauensfaktors (Vertrauen in diese Schätzung, siehe Tabelle 10, S. 37)

Tabelle 61: Generierungstabelle für ein Interaktionsszenario

Zentrale Elemente eines Interaktionsszenarios sind Stimulus und Response; ein Stimulus kann mehrere Responses haben; es können also verschiedene Interaktionsszenarios mit dem gleichen Stimulus vorkommen.

Die Angaben zur Response-Prüfung verweisen auf verschiedene Evaluationsmethoden, da es viele Möglichkeiten zur Evaluation der Response eines Artefakts auf den Stimulus gibt[1]. Nach ihrer Einbeziehung von Benutzern kategorisiert, sind Inspektionen durch Experten (A für Analyse) und Evaluationen durch Endbenutzer (U für User-Evaluation) angegeben. Bei User-Evaluationen werden Usability-Probleme durch Anwendungsfehler und menschliche Fehler aufgedeckt. Bei Inspektionen durch Experten wird das Design des User Interfaces, der Interaktionen oder der Softwarearchitektur analysiert und evaluiert, um Ursachen und Indikatoren für mögliche Usability-

1 Einen Überblick über Usability-Evaluationsmethoden aus softwaretechnischer Perspektive bieten [FJM05].

Probleme offen zu legen. Die Interaktionsszenarios werden von einem oder mehreren Patterns abgeleitet (Extraktionsmethode siehe Abschnitt 3.4.1). Ein Pattern wird referenziert, damit seine Inhalte während der Evaluation des betreffenden Interaktionsszenarios nachgelesen werden können. Damit die Abstraktionsebene der Interaktionsszenarios erhalten bleibt, werden nicht Lösungen, sondern die im Pattern beschriebene Anforderung in ein bis zwei Sätzen kurz zusammengefasst.

A.2 KATALOG DER INTERAKTIONSSZENARIOS

Ein erster Katalog der Interaktionsszenarios wurde 2009 von Stefan Seitz im Rahmen seiner Diplomarbeit erstellt [Sei09]. Diese Ergebnisse wurden auf der *MobiWIS 2011* mit Stefan Seitz und Volker Gruhn veröffentlicht [BSG11]. Katharina König übersetzte die Interaktionsszenarios vom Englischen ins Deutsche; anschließend wurden sie von der Autorin dieser Arbeit vervollständigt und von Marion Wiesner korrigiert.

Folgende Interaktionskategorien entstanden durch das Clustern von Interaktionsszenarios [Sei09]. Die Nummern in den Klammern geben die Anzahl der Interaktionsszenarios an, die in einer Interaktionskategorie enthalten sind; ein Interaktionsszenario kann auch in mehreren Kategorien vorkommen. Stand der Datenbank ist Juli 2012[2].

I_1. Erfassung und Behandlung von Benutzereingaben (13)

I_2. Orientierung (3)

I_3. Navigieren, Browsen, Auswählen (3)

I_4. Ausführen, Wiederholen und Zurücknehmen von Befehlen (9)

I_5. Interagieren mit unbekannten Systemen (3)

I_6. Selektion und Exploration von Daten (3)

I_7. Bearbeiten von Grafiken und grafischen Objekten (0)

I_8. Austausch und Manipulation von Daten (5)

I_9. Informationsbeschaffung (2)

I_{10}. Strukturieren und Anzeigen von Content, Informationen, Daten (11)

I_{11}. Behandlung von Fehlern und Hilfe (9)

I_{12}. Anpassen durch Benutzer, für Benutzer, für Aufgaben (12)

I_{13}. Kooperation (in einem System, mit anderen Systemen) (7)

2 Eine frühere Version ist verfügbar unter http://cassini.saturn-lab.com/

Name	*Beschreibung*
Nummer	1
Kurzname	System-Feedback (System Feedback)
Quelle	Nutzer
Stimulus	möchten über Status, Aktivitäten und Veränderungen des Systems informiert bleiben.
Response	Das System liest die vom Nutzer angeforderten Daten aus und zeigt sie entsprechend der menschlichen Bedürfnisse und Fähigkeiten an.
Response-Prüfung	Nutzer erhalten das Feedback zeitnah bzw. sofort. (A/U) Nutzer erhalten alle notwendigen Daten. (A/U) Nutzer nehmen alle für sie notwendigen Daten wahr. (U) Die Daten sind aktuell. (A/U)
Usability-Pattern(s)	[BJK01] 17. Observing System State: Ein Nutzer könnte wollen, dass ihm die Systemstatusdaten, die notwendig sind, damit ein System läuft, angezeigt werden. Systemdesigner sollten menschliche Bedürfnisse und Fähigkeiten berücksichtigen, wenn sie darüber entscheiden, welche Aspekte eines Systemstatus sie anzeigen und wie sie diese anzeigen lassen. [FGB03] 17. System Feedback: Teile dem Nutzer Veränderungen des Systems mit.
Interaktions-kategorien	Nutzereingaben erfassen und behandeln
Usability-Ziel(e)	(+) Sicherheit: Benutzer erhält Informationen, was Aktivitäten bewirken und hat das Gefühl, das System zu kontrollieren (+) Erlernbarkeit: Lernen, wie das System auf eine Aktion reagiert
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 62: Interaktionsszenario System-Feedback

Name	*Beschreibung*
Nummer	2
Kurzname	Aktionen auf mehreren Objekten (Actions on Multiple Objects)
Quelle	Benutzer
Stimulus	möchten mit einer Auswahl von Objekten eine oder mehrere Aktionen durchführen.
Response	Nutzer setzen eine Reihe von Daten zusammen, bestimmen mit den Daten durchzuführende Aktionen und starten die Ausführung.
Response-Prüfung	Die Aktion wird auf dem (den) ausgewählten Objekt(en) ausgeführt. (A) Das Zusammenstellen von Objekten und Ausführen einer Aktion auf dieser Gruppe von Objekten ist weniger zeitaufwendig als einzelne Objekte individuell zu bearbeiten. (U, A)
Usability-Pattern(s)	[BJK01] 1. Aggregating Data: Ein Nutzer könnte eine oder mehr als eine Aktion auf einem oder mehreren Objekten ausführen wollen. Systeme sollten Nutzern erlauben, willkürliche Kombinationen von Daten auszuwählen und mit ihnen zu arbeiten. [FGB03] 18. Actions for Multiple Objects: Stelle einen Mechanismus zur Verfügung, der es dem Nutzer erlaubt, Aktionen individuell anzupassen oder zu aggregieren. [FGB03] 22. Scripting: Stelle einen Mechanismus zur Verfügung, der es dem Nutzer erlaubt, eine Reihe von Befehlen oder Aktionen mit verschiedenen Objekten durchzuführen.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen Austausch und Manipulation von Daten
Usability-Ziel(e)	(+) Effizienz: Eine Aktion auf einer Gruppe von Elementen auszuführen, geht schneller als auf einzelnen Elementen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 63: Interaktionsszenario Aktionen auf mehreren Objekten (Actions on Multiple Objects)

Name	*Beschreibung*
Nummer	3
Kurzname	Makros (Macros)
Quelle	Benutzer
Stimulus	möchten eine Reihe von Befehlen ausführen, ohne die einzelnen Aufgaben starten zu müssen und ohne den gesamten Prozess überwachen zu müssen.
Response	Das System liest im Voraus alle für den Vorgang notwendigen Nutzereingaben ein und führt sie selbständig durch.
Response-Prüfung	Die Sequenz von Befehlen wird ausgeführt. (A) Während der Ausführung sind keine Aktivitäten seitens der Benutzer notwendig. (A)
Usability-Pattern(s)	[Tido5] 53 Macros: Makros sind Aktionen, die aus mehreren Einzelaktionen bestehen. Benutzer können sie selbst als Folgen von Einzelaktionen zusammenstellen. [BJK01] 2. Aggregating Commands: Ein Nutzer könnte eine lange laufende, vielschrittige Prozedur ausführen wollen, die aus verschiedenen Kommandos besteht. Systeme sollten ein Batch oder Makros zur Verfügung stellen, damit Nutzer Kommandos aggregieren können.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effizienz: Automatische und damit sehr schnelle Bearbeitung von häufig auftretenden Interaktionen, die aus mehreren einzelnen Interaktionen bestehen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 64: Interaktionsszenario Makros (Macros)

Name	*Beschreibung*
Nummer	4
Kurzname	Abbrechen (Cancel)
Quelle	Benutzer
Stimulus	haben eine Operation gestartet und möchten sie stoppen und widerrufen.
Response	Das System beendet die Ausführung, sobald der Nutzer die Abbrechen-Funktion aktiviert. Der vor dem Starten des Vorgangs bestehende Systemstatus wird wieder hergestellt.
Response-Prüfung	Das Abbrechen wird innerhalb einer bestimmten Zeitspanne durchgeführt. (A, U) Der Prozess des Abbrechens wird gegenüber dem Benutzer kommuniziert. (A, U)
Usability-Pattern(s)	[Tid05] 50. Cancelability (Abbruchsmöglichkeit): Biete eine Möglichkeit an, um eine zeitaufwendige Aktion abzubrechen, ohne dass Nebenwirkungen auftreten. [BJK01, GJB05] 3. Canceling Commands: Ein Nutzer ruft eine Funktion auf und möchte dann, dass die Funktion nicht länger ausgeführt wird. Systeme sollten dem Nutzer erlauben, Abläufe abzubrechen. [FGB03] 19. Cancel: Erlaube dem Nutzer beispielsweise zur Verhinderung von Fehlern, einen Befehl abzubrechen, der aufgerufen, aber noch nicht abgeschlossen wurde.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effizienz: Nach einem Fehler ist das System schnell wieder einsatzfähig. (+) Effektivität: Das System arbeitet zuverlässig und ermöglicht die Weiterbearbeitung der Aufgabe des Benutzers. (+) Sicherheit: Die Daten der Benutzer sind sicher (die Daten, mit denen sie gearbeitet haben, Arbeitsergebnisse). (+) Erlernbarkeit: Lernen durch Versuch und Irrtum möglich.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 65: Interaktionsszenario Abbrechen (Cancel)

Name	*Beschreibung*
Nummer	5
Kurzname	Nachsichtiges Formular (Forgiving Format)
Quelle	System
Stimulus	bietet den Nutzern ein Formular zur Datenabfrage an.
Response	Nutzer geben die Daten in ein Feld ein und müssen dabei kein besonderes Format und keine besondere Syntax berücksichtigen. Das System interpretiert die Eingaben selbständig.
Response-Prüfung	Wo technisch möglich, erkennt das System den Typ und das Format der eingegebenen Daten automatisch und korrekt. (A)
Usability-Pattern(s)	[Tido5] 68. Forgiving Format: Erlaube Nutzern, Texte verschiedenen Formats und unterschiedlicher Syntax einzugeben. Erstelle die Anwendung so, dass sie den Text intelligent interpretieren kann.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Benutzer müssen sich nicht über Formate Gedanken machen, sondern können das Formular ohne langes Nachdenken ausfüllen. (+) Nützlichkeit: Auch Benutzergruppen ohne technische Erfahrung können das System einfach benutzen.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 66: Interaktionsszenario Nachsichtiges Formular (Forgiving Format)

Name	*Beschreibung*
Nummer	6
Kurzname	Strukturiertes Format (Structured Format)
Quelle	System
Stimulus	bietet den Nutzern ein Formular zur Datenabfrage an.
Response	In dem automatisch erstellten Formular spiegelt sich das Format der angeforderten Daten in der Struktur des Formulars und der Eingabefelder wider.
Response-Prüfung	Das Formular wird anhand des benötigten Datenformats korrekt strukturiert. (A) Benutzer erkennen das benötigte Format der Daten. (U - Experten oder Evaluation der Benutzung)
Usability-Pattern(s)	[Tido5] 69. Structured Format: Verwende an Stelle eines Textfeldes mehrere Textfelder, so dass die Struktur der angefragten Daten widergespiegelt werden kann.
Interaktions-kategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Benutzer können das Formular ohne langes Nachdenken über sinnvolle Eingabeformate ausfüllen. (+) Nützlichkeit: Auch Benutzergruppen ohne technische Erfahrung können das System einfach benutzen.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 67: Interaktionsszenario Strukturiertes Format (Structured Format)

Name	*Beschreibung*
Nummer	7
Kurzname	Eingabefenster (Input Widgets)
Quelle	System
Stimulus	bietet den Nutzern zur Datenabfrage ein Formular an.
Response	Um spezielle Datentypen unter minimaler Verwendung der Tastatur einzugeben, werden mehrere Felder oder Auswahl-Widgets verwendet.
Response-Prüfung	Die verwendeten Widgets und die übertragenen Daten stimmen mit dem benötigten Eingabeformat überein. (A) Die Benutzung der Tastatur bzw. des Keypads wird reduziert. (U)
Usability-Pattern(s)	[Lit10] 15. Text Entry: Editieren auf der Stelle ermöglicht eine Eingabe mitten im aktuellen Kontext, in einem Teil des Bildschirmes. Vollbild-Editieren, bei dem das Eingabefeld den ganzen Bildschirm überdeckt, sollte auf Wunsch möglich sein.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Benutzer können ein Formular durch die Formatierung zügig ausfüllen.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 68: Interaktionsszenario Eingabefenster (Input Widgets)

Name	*Beschreibung*
Nummer	8
Kurzname	Gleiche oder Separate Ansicht (Same or Separate Screen, Pagination)
Quelle	System
Stimulus	bietet den Nutzern ein Formular zur Datenabfrage an.
Response	Abhängig von der Menge des einzugebenden Textes findet die Eingabe (automatisch) direkt oder in einer separaten Ansicht (Bildschirm, Seite) statt.
Response-Prüfung	Die Größe des Eingabefelds stimmt mit der Menge von benötigten Daten überein. (A)
Usability-Pattern(s)	[Lit10] Pagination: Beim Erstellen eines Systems mit sehr vielen Informationen ist es möglich, diese in einzelne Seiten wie bei einem Buch aufzuteilen.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Das Aufteilen von Formularen, die sonst wegen der Eingabe einer großen Datenmenge über mehrere Seiten laufen würden, ermöglicht eine Minimierung des Scrollens.
Pot. SA-sensitivität	2**: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere weitere Beispiele zeigen, dass die Einschätzung grundsätzlich zutrifft.

Tabelle 69: Interaktionsszenario Gleiche oder Separate Ansicht (Same or Separate Screen, Pagination)

Name	*Beschreibung*
Nummer	9
Kurzname	Lückenfüller (Fill-in-the-Blanks)
Quelle	System
Stimulus	möchte dem Nutzer die Bedeutung oder den Zweck eines Formularfeldes ohne formale Erklärung vermitteln.
Response	Innerhalb eines Satzes oder einer Aussage werden leere Eingabefelder mit dem entsprechenden Format angezeigt. Diese Lücken sind von Benutzern zu füllen.
Response-Prüfung	Die Eingabefelder entsprechen dem gewünschten Format des Datentyps. (A) Benutzer erkennen die Bedeutung des Eingabefeldes innerhalb einer bestimmten Zeitspanne. (U) Die von den Benutzern eingegebenen Daten stimmen mit der Art der eingeforderten Information überein. (U)
Usability-Pattern(s)	[Tido5] 70. Fill-in-the-Blanks: Ordne eins oder mehrere Felder in Form eines Satzes oder einer Aussage an, wobei die Felder Leerstellen sind, die vom Nutzer ausgefüllt werden müssen.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Der Kontext ermöglicht das schnelle Erkennen des nötigen Eingabeformats. (+) Erlernbarkeit: Benutzer erkennen schnell, welches Eingabeformat gefordert ist.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 70: Interaktionsszenario Lückenfüller (Fill-in-the-Blanks)

Name	*Beschreibung*
Nummer	10
Kurzname	Vor-Ort-Hinweis (Hint-in-Place)
Quelle	System
Stimulus	möchte dem Nutzer die Bedeutung oder den Zweck eines Formularfeldes ohne formale Erklärung vermitteln.
Response	Ein direkt beim/im Eingabefeld angezeigter Hinweis oder ein Beispiel helfen dem Nutzer zu verstehen, was er eingeben muss.
Response-Prüfung	Es wird automatisch ein Hinweis oder ein Beispiel angegeben. (A) Benutzer erkennen die Bedeutung des Eingabefeldes innerhalb einer bestimmten Zeitspanne. (U) Die von den Benutzern eingegebenen Daten stimmen mit der Art der eingeforderten Information überein. (U)
Usability-Pattern(s)	[Tid05] 71. Input Hints: Platziere neben ein leeres Textfeld einen Satz oder ein Beispiel, das erklärt, was verlangt wird. [Tid05] 72. Input Prompt: Befülle ein Textfeld oder Dropdown im Voraus mit einer Eingabeaufforderung, die den Nutzer darauf hinweist, was er tun oder eingeben soll.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Benutzer erfassen schnell, was sie eingeben müssen.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 71: Interaktionsszenario Vor-Ort-Hinweis (Hint-in-Place)

Name	*Beschreibung*
Nummer	11
Kurzname	Gute Standardeinstellungen (Good Defaults)
Quelle	System
Stimulus	möchte möglichst wenig Daten manuell eingeben.
Response	Die Eingabefelder wurden vorab automatisch mit Werten gefüllt, von denen das System annimmt, dass sie am wahrscheinlichsten eingegeben oder ausgewählt werden. Sie sind editierbar.
Response-Prüfung	Die Felder werden mit logisch nachvollziehbaren Eingaben gefüllt. (A) Die vom System vorgeschlagen Werte werden häufig von den Benutzern verwendet und nicht überschrieben. (U)
Usability-Pattern(s)	[Tido5] 77. Good Defaults: Wenn angemessen, fülle Formularfelder nach bestem Ermessen mit den Werten aus, die der Nutzer vermutlich eingeben wird.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Wenn die Voreinstellungen korrekt sind, können die Benutzer sofort zur nächsten Aufgabe übergehen. Zudem wird die manuelle Dateneingabe reduziert.
Pot. SA-sensitivität	2*: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 72: Interaktionsszenario Gute Voreinstellungen (Good Defaults)

Name	*Beschreibung*
Nummer	12
Kurzname	Auto-Komplettierung (Auto-Completion)
Quelle	System
Stimulus	möchte möglichst wenig Daten manuell eingeben.
Response	Während Nutzer Text eingeben, empfiehlt das System einzugebende Daten und bietet dem Nutzer diese zur Auswahl an.
Response-Prüfung	Die vom System vorgeschlagenen Werte werden mit einer hohen Häufigkeit von den Benutzern ausgewählt und eingegeben. (U)
Usability-Pattern(s)	[Tid05] 73. Autocompletion: Nimm, noch während der Benutzer Daten in ein Textfeld eingibt, die möglichen Antworten vorweg und vervollständige den Eintrag. [Lit10] 25. Autocomplete (in Text Entry): Rate, was eine Person eintragen will und biete an, den Text zu vervollständigen.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Wenn die vorgeschlagenen Daten korrekt sind, können Benutzer das Formular schnell ausfüllen. Manuelle Dateneingabe wird reduziert.
Pot. SA-sensitivität	3**: Unvorhersehbare Modifikationen. Mehrere weitere Beispiele zeigen, dass die Einschätzung grundsätzlich zutrifft.

Tabelle 73: Interaktionsszenario Auto-Komplettierung (Auto-Completion)

Name	*Beschreibung*
Nummer	13
Kurzname	Sofortige Gültigkeitsprüfung (Instant Validation)
Quelle	System
Stimulus	möchte sicherstellen, dass die vom Nutzer eingegebenen Daten valide und korrekt sind.
Response	Während oder nachdem die Daten eingegeben oder geändert werden, prüft das System die Gültigkeit der Eingaben und teilt dem Benutzer Fehler und Verbesserungsvorschläge mit.
Response-Prüfung	Die Überprüfung erfolgt innerhalb einer bestimmten Zeitspanne. (A) Fehler werden schon während der Eingabe der Daten erkannt. (A) Fehler werden durch das System erkannt und verständlich kommuniziert. (U)
Usability-Pattern(s)	[Lit10] 15. Text Entry, Passive and Active Validation: Aktive Validierung erfolgt nach jedem Zeichen, das der Benutzer eingibt. Passive Validierung erfolgt nach der vollständigen Eingabe, entweder weil die Bedeutung der Eingabe erst dann möglich ist oder der Server die Daten nur so verarbeiten kann.
Interaktionskategorien	Behandlung von Fehlern und Hilfe Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Die sofortige Kontrolle ermöglicht eine zügige Fehlerkorrektur. Damit werden spätere Verzögerungen durch Fehler und die damit verbundene erneute Dateneingabe (Wiederholung einer Tätigkeit) vermieden. (+) Sicherheit: Kleine Fehler können sofort behoben werden, der gesamte Datensatz wird sofort korrekt eingetragen. (+) Erlernbarkeit: Die Eingaben in Formulare werden sofort einzeln validiert, der Benutzer erhält eine sofortige Rückmeldung.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 74: Interaktionsszenario Sofortige Gültigkeitsprüfung (Instant Validation)

Name	*Beschreibung*
Nummer	14
Kurzname	Prüfen auf Korrektheit (Checking for Correctness)
Quelle	System
Stimulus	möchte sicherstellen, dass die vom Nutzer eingegebenen Daten valide und korrekt sind.
Response	Sobald der Nutzer das Formular abschickt, überprüft das System die Eingaben auf Fehler. Wenn es Fehler gibt, startet das System eine Fehlerbehandlungsroutine.
Response-Prüfung	Eingabefehler werden durch das System erkannt. (A) Durch das System erkannte Eingabefehler werden behandelt. (A)
Usability-Pattern(s)	[BJK01] 5. Checking for Correctness: Ein Nutzer könnte einen Fehler machen, den er oder sie nicht bemerkt. Systeme sollten eine Korrektur vorschlagen oder durchführen. [FGB03] 20. Data Validation: Verifiziere, ob die Eingaben in den Formularfeldern korrekt sind.
Interaktionskategorien	Behandlung von Fehlern und Hilfe Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effektivität: Das System unterstützt die Benutzer bei der korrekten Durchführung ihrer Aufgaben. (+) Sicherheit: Die gespeicherten Daten der Nutzer sind in dem Maße korrekt und valide, wie das System die Prüfung durchführt. Es werden bestimmte Fehler vermieden.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 75: Interaktionsszenario Prüfen auf Korrektheit (Checking for Correctness)

Name	*Beschreibung*
Nummer	15
Kurzname	Aufforderung zur Fehlerbehebung (Prompt for Error Correction)
Quelle	System
Stimulus	erkennt Fehler in den von Benutzern ausgefüllten und abgeschickten Formularen.
Response	Die Nutzer werden direkt vor Ort auf die Fehler hingewiesen, erhalten Verbesserungsvorschläge und können ihre Fehler daraufhin korrigieren.
Response-Prüfung	Benutzer nehmen die Informationen zum Fehler wahr. (U) Benutzer beheben ihre Fehler selbständig. (U)
Usability-Pattern(s)	[Tid05] 78. Same-Page Error Messages: Platziere Fehlermeldungen direkt auf der Seite mit dem Formular. Markiere den oberen Rand der Seite mit einer Fehlermeldung und, wenn möglich, positioniere Indikatoren neben den entstehenden Bedienelementen.
Interaktionskategorien	Behandlung von Fehlern und Hilfe Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effektivität: Das System unterstützt die Benutzer bei der korrekten Durchführung ihrer Aufgaben. (+) Sicherheit: Die gespeicherten Daten der Nutzer sind in dem Maße korrekt und valide, wie das System die Prüfung durchführt. Es werden bestimmte Fehler vermieden.
Pot. SA-sensitivität	2*: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 76: Interaktionsszenario Aufforderung zur Fehlerbehebung (Prompt for Error Correction)

Name	*Beschreibung*
Nummer	16
Kurzname	Leistungsabhängige Listenlänge (Result List Length)
Quelle	System
Stimulus	möchte die Ergebnisse einer Anfrage anzeigen.
Response	Die Anzahl von Suchergebnissen wird durch die Länge und die Ladezeit einer Seite bestimmt. Die Seite wird, sofern gewünscht, in bestimmten Zeitabständen aktualisiert.
Response-Prüfung	Die Liste der besten Suchergebnisse wird auf dem Endgerät angezeigt und in bestimmten Zeitabständen aktualisiert. (A) Die Benutzer finden das von ihnen Gesuchte in den angebotenen Ergebnissen direkt auf der ersten Seite. (U)
Usability-Pattern(s)	[Lit10] 6. Returned Results: Zeige genau einen Bildschirm voller Ergebnisse an. Biete eine Navigation zu weiteren Ergebnissen über Links an, die zur Vermeidung von Scrollen unten angeordnet werden.
Interaktionskategorien	Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effizienz: Nicht alle Ergebnisse werden ermittelt und danach übertragen, sondern nur ein Teil davon. Die Response-Zeit ist kürzer. Die übertragenen Daten sind vom Umfang her geringer, so dass die Latenzzeit durch die Dauer der Übertragung gering ist.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 77: Interaktionsszenario Leistungsabhängige Listenlänge (Result List Length)

Name	*Beschreibung*
Nummer	17
Kurzname	Befehlsprotokoll (Command History)
Quelle	Benutzer
Stimulus	möchten wissen, welche Auswahlen getroffen wurden und welche Aktionen begonnen oder durchgeführt wurden (durch sie selbst oder das System).
Response	Einzelne Selektionen und Aktionen werden in einer Historie protokolliert. Die Benutzer greifen auf eine Aufzeichnung aller vorher getätigten Auswahlen und Aktionen zu.
Response-Prüfung	Die Protokollierung ist komplett und aktuell und erfordert nicht zu viel Leistung. (A) Das Protokoll enthält und präsentiert für Benutzer verständliche Informationen. (U)
Usability-Pattern(s)	[Tid05] 52. Command History: Wenn der Nutzer Aktionen ausführt, zeichne sichtbar auf, was er wann und wofür getan hat. [FGB03] 21. History Logging: Erstelle ein Log der Aktionen des Nutzers (und möglichst vom System), um zurück zu verfolgen, was getan wurde.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen Orientierung
Usability-Ziel(e)	(+) Sicherheit: Die Benutzer können Benutzer- oder Systemfehler (oder beispielsweise auch Angriffe auf das System) erkennen und zurückverfolgen. (-) Effizienz: Die Protokollierung kann die Effizienz des Systems verringern und damit die Effizienz der Bearbeitung der Aufgabe.
Pot. SA-sensitivität	2**: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere weitere Beispiele zeigen, dass die Einschätzung grundsätzlich zutrifft.

Tabelle 78: Interaktionsszenario Befehlsprotokoll (Command History)

Name	*Beschreibung*
Nummer	18
Kurzname	Alternativansichten (Alternative Views)
Quelle	Benutzer
Stimulus	möchten, dass das System ihren Bedürfnissen und Vorlieben, ihren Rollen und Aufgaben genügt.
Response	Das System bietet für unterschiedliche Nutzer und Verwendungszwecke eine Auswahl von Ansichten und/oder Interfaces an, zwischen denen gewechselt werden kann.
Response-Prüfung	Die gleichen Operationen werden durchgeführt, auch wenn sie von verschiedenen User Interfaces aus gestartet werden. (A) Die von einem Benutzer ausgewählte Ansicht bzw. das Interface entspricht seinen Bedürfnissen und Präferenzen bzw. unterstützt ihre aktuelle Rolle und die daraus folgende(n) Aufgabe(n). (U)
Usability-Pattern(s)	[Tid05] 16. Alternative Views: Lasse Nutzer aus der Standardsicht heraus zwischen alternativen Sichten wählen, die sich dem Aussehen nach und strukturell unterscheiden. [BJK01] 26. Supporting Visualization: Ein Nutzer möchte Daten aus verschiedenen Perspektiven betrachten. Abhängig von der vom Nutzer gerade durchgeführten Aufgabe sollte das System verschiedene Zusatzansichten anbieten. [FGB03] 23. Multiple Views: Biete verschiedenen Nutzern verschiedene Ansichten für verschiedene Aufgaben an. [FGB03] 29. Workflow Model: Stelle verschiedenen Nutzern nur die Werkzeuge oder Aktionen zur Verfügung, die für konkrete Aufgaben bezüglich eines Ausschnittes von Daten nötig sind.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben
Usability-Ziel(e)	(+) Effizienz: Schnelle Aufgabenbewältigung ist möglich, da die wichtigen Interaktionen im User Interface gut sichtbar und schnell verfügbar sind. (+) Nützlichkeit: Die vom System angebotenen Funktionen sind auf verschiedene Benutzergruppen abgestimmt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 79: Interaktionsszenario Alternativansichten (Alternative Views)

Name	*Beschreibung*
Nummer	19
Kurzname	Modi und Profile (Modes and Profiles)
Quelle	Benutzer
Stimulus	möchten, dass das System ihren Bedürfnissen und Vorlieben, ihren Rollen und Aufgaben genügt.
Response	Das System erstellt, modifiziert, nutzt Profile oder Modi zur Speicherung der Präferenzen, Rollen und Aufgaben der Nutzer.
Response-Prüfung	Die gleichen Operationen werden durchgeführt, auch wenn sie von verschiedenen User Interfaces aus gestartet werden. (A) Benutzer konfigurieren ein Profil oder einen Modus, um das System ihren Vorlieben, ihrer aktuellen Rolle und/oder ihren Aufgaben entsprechend anzupassen. (U)
Usability-Pattern(s)	[FGB03] 26. User Modes: Stelle verschiedene Modi zur Verfügung, die die unterschiedlichen Funktionsumfänge bieten, die ein bestimmter Nutzertyp oder derselbe Nutzer benötigt, um verschiedene Aufgaben auszuführen. [FGB03] 27. User Profiles: Erfasse und erstelle ein Profil von jedem Nutzertyp, damit bestimmte Attribute des Systems, z. B. das Layout der Benutzerschnittstelle, die zu zeigende Datenmenge oder die Optionen, für verschiedene Nutzer neu gesetzt werden können. [FGB03] 29. Workflow Model: Stelle verschiedenen Nutzern nur die Werkzeuge oder Aktionen zur Verfügung, die für konkrete Aufgaben bezüglich eines Ausschnittes von Daten nötig sind.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben
Usability-Ziel(e)	(+) Effizienz: Schnelle Aufgabenbewältigung ist möglich, da die wichtigen Interaktionen im User Interface gut sichtbar und schnell verfügbar sind. (+) Nützlichkeit: Die vom System angebotenen Funktionen sind auf verschiedene Benutzergruppen abgestimmt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 80: Interaktionsszenario Modi und Profile (Modes and Profiles)

Name	*Beschreibung*
Nummer	20
Kurzname	Internationalisierung (Supporting International Use)
Quelle	Benutzer
Stimulus	möchten, dass das System ihren Bedürfnissen und Vorlieben, ihren Rollen und Aufgaben genügt.
Response	Das System wird so konfiguriert, dass Benutzer ihre bevorzugte Sprache verwenden können und weitere Aspekte ihrer Kultur berücksichtigt werden.
Response-Prüfung	Die Umstellung erfolgt in einer bestimmten Zeitspanne. (A) Benutzer können das Systems konfigurieren (oder das System konfiguriert sich selbst automatisch). (U)
Usability-Pattern(s)	[BJK01] 12. Supporting International Use: Ein Nutzer könnte eine Anwendung konfigurieren wollen, um in ihrer oder seiner Sprache zu kommunizieren bzw. entsprechend den Normen ihrer oder seiner Kultur.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben
Usability-Ziel(e)	(+) Effizienz: Die Benutzer können schneller mit dem System interagieren, da sie Texte in ihrer eigenen Sprache schneller verstehen. (+) Nützlichkeit: Benutzergruppen verschiedener Kulturkreise werden unterstützt. (+) Erlernbarkeit: Die Benutzer können schneller verstehen, wie sie mit dem System interagieren müssen, wenn es in ihrer eigenen Sprache Informationen anbietet.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 81: Interaktionsszenario Internationalisierung (Supporting International Use)

Name	*Beschreibung*
Nummer	21
Kurzname	Mehrkanalzugang (Multi-Channel Access)
Quelle	Benutzer
Stimulus	möchten Zugang zum System mithilfe einer Hardware ihrer Wahl erhalten (und damit Zugang für die jeweiligen Eingabe- und Ausgabemöglichkeiten).
Response	Das Eingabe-/Ausgabe-Gerät eines Benutzers wird mit der Hardware des Systems verbunden und funktioniert mit der Anwendung.
Response-Prüfung	Das jeweilige Endgerät funktioniert sofort, ohne separate Konfiguration. (A) Die Benutzerschnittstelle sieht gut aus. (A, U)
Usability-Pattern(s)	[FGB03] 33. Multi-Channel Access: Ein Nutzer greift auf ein System mit verschiedenen Geräten (mobil, Desktop) zu. Stelle einen Mechanismus zur Verfügung, der mehrere Kanäle anbietet, die bestimmte Geräte jeweils besonders unterstützen.
Interaktionskategorien	Strukturieren und Anzeigen von Content, Informationen, Daten Kooperation (in einem System, mit anderen Systemen)
Usability-Ziel(e)	(+) Nützlichkeit: Bestimmte Benutzergruppen werden gesondert unterstützt – beispielsweise Personen mit Handicap oder Personen mit neuen mobilen Endgeräten.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 82: Interaktionsszenario Mehrkanalzugang (Multi-Channel Access)

Name	*Beschreibung*
Nummer	22
Kurzname	Mehrfach-Widerruf (Multi-Level Undo)
Quelle	Benutzer
Stimulus	möchten das Ergebnis einer Aktion oder Wahl rückgängig machen und zu einem vorhergehenden Status zurückkehren.
Response	Nutzer gehen zum gewünschten Punkt der Aktions-/Auswahlchronik zurück. Dadurch setzen sie alle Aktionen und Auswahlen bis zu diesem Punkt zurück.
Response-Prüfung	Der ausgewählte Systemstatus ist in einer bestimmten Zeitspanne wieder hergestellt. (A) Das System ist stabil. (A) Der wieder hergestellte Systemstatus ist genau derjenige, den die Benutzer ausgewählt hatten. (U)
Usability-Pattern(s)	[Tid05] 51. Multi-Level Undo: Biete die Möglichkeit an, mehrere aufeinander abfolgende Aktionen eines Benutzers rückgängig zu machen. [BJK01] 21. Supporting Undo: Ein Nutzer führt einen Arbeitsschritt aus, möchte Auswirkungen aufheben. Das System sollte es ermöglichen, zu dem Systemzustand vor Ausführung des Arbeitsschrittes zurückzukehren. Mehrfach. [FGB03] 32. Multi-Level Undo: Nutzer führen Aktionen aus, die sie später rückgängig machen wollen, weil sie z. B. einen Fehler gemacht oder ihre Meinung geändert haben. Halte eine Liste der getätigten Nutzeraktionen vor, erlaube es dem Nutzer, Aktionen rückgängig zu machen.
Interaktions-kategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effizienz: Schnelle automatische Korrektur erfordert keine manuelle Arbeit. (+) Effektivität: Zielerreichung trotz Fehlern. (+) Sicherheit: Fehler ohne Auswirkungen auf die Daten der Benutzer. (+) Erlernbarkeit: Lernen durch Versuch und Irrtum.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 83: Interaktionsszenario Mehrfach-Widerruf (Multi-Level Undo)

Name	*Beschreibung*
Nummer	23
Kurzname	Workflow-Modell (Workflow Model)
Quelle	Benutzer
Stimulus	erfüllen einen einzelnen Schritt in einer Workflow-Kette auf einem festgelegten Teil der Daten.
Response	Es werden die Werkzeuge und Aktionen zur Verfügung gestellt, die benötigt werden, um die konkrete Aufgabe zu erfüllen.
Response-Prüfung	Benutzer führen ihre Arbeit mit den angebotenen Werkzeugen und Aktionen aus. (U)
Usability-Pattern(s)	[FGB03] 29. Workflow Model: Stelle verschiedenen Nutzern nur die Werkzeuge oder Aktionen zur Verfügung, die für konkrete Aufgaben bezüglich eines Ausschnittes von Daten nötig sind.
Interaktions-kategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Austausch und Manipulation von Daten
Usability-Ziel(e)	(+) Effizienz: Schnelle Aufgabenbewältigung ist möglich, da die wichtigen Interaktionen im User Interface gut sichtbar und schnell verfügbar sind. (+) Sicherheit: Nur die notwendigen Befehle werden angeboten, somit sind unerwünschte Datenmanipulationen nicht (oder zumindest erschwert) möglich. (+) Nützlichkeit: Die vom System angebotenen Funktionen sind auf verschiedene Benutzergruppen abgestimmt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 84: Interaktionsszenario Workflow-Modell (Workflow Model)

Name	*Beschreibung*
Nummer	24
Kurzname	Konfliktfreie Nebenläufigkeit (Non-conflicting Application Concurrency)
Quelle	Benutzer
Stimulus	möchten mit mehreren Anwendungen gleichzeitig arbeiten.
Response	Das System kann gleichzeitig mit anderer Software betrieben werden, ohne dass es zu Konflikten wegen Nebenläufigkeit kommt.
Response-Prüfung	Das System ist robust und fehlertolerant. (A)
Usability-Pattern(s)	[BJK01] 4. Using Applications Concurrently: Ein Nutzer könnte mit einer beliebigen Kombination von Anwendungen gleichzeitig arbeiten. Systeme sollten gewährleisten, dass Nutzer mehrere Anwendungen verwenden können, ohne dass diese in Konflikt miteinander geraten.
Interaktionskategorien	Kooperation (in einem System, mit anderen Systemen)
Usability-Ziel(e)	(+) Effizienz: Keine Verzögerung durch den Start benötigter Programme. (+) Sicherheit: Sicherung der Daten vor unerwünschter Manipulation aufgrund von Fehlern.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 85: Interaktionsszenario Konfliktfreie Nebenläufigkeit (Non-conflicting Application Concurrency)

Name	*Beschreibung*
Nummer	25
Kurzname	Geräteunabhängigkeit (Device Independence, ehem. Non-conflicting Device Usage)
Quelle	Benutzer
Stimulus	möchten ein neues Endgerät installieren und verwenden.
Response	Selbst wenn Probleme durch auftretende Konflikte mit dem Gerät aufkommen, können Benutzer das Gerät zum Interagieren mit der Anwendung verwenden.
Response-Prüfung	Die Häufigkeit und von Geräte- und Software-Konflikten bleibt unter festgelegten Grenzwerten (beispielsweise die Anzahl von Konflikten pro Zeiteinheit, die Gesamtanzahl von Konflikten). (A) Das System ist fehlertolerant, so dass Auswirkungen von Geräte- und Softwarekonflikten gering sind. (A)
Usability-Pattern(s)	[BJK01] 6. Maintaining Device Independence: Ein Nutzer versucht ein neues Gerät zu installieren. Systeme sollten so gestaltet sein, dass die Schwere und Häufigkeit von Gerätekonflikten reduziert wird. Wenn Gerätekonflikte auftreten, sollte das System die notwendigen Informationen bereitstellen, um das jeweilige Problem zu lösen oder nach Hilfe suchen.
Interaktions-kategorien	Behandlung von Fehlern und Hilfe Kooperation (in einem System, mit anderen Systemen)
Usability-Ziel(e)	(+) Sicherheit: Die Fehlertoleranz gegenüber Gerätekonflikten schützt die Daten der Benutzer.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 86: Interaktionsszenario Geräteunabhängigkeit (Device Independence, ehem. Non-conflicting Device Usage)

Name	*Beschreibung*
Nummer	26
Kurzname	Wiederherstellung nach Betriebsausfall (Recovering From Failure)
Quelle	System
Stimulus	arbeitet fehlerhaft oder stürzt ab.
Response	Das System minimiert die Menge an aufgrund des Defekts verloren gehenden Daten oder stellt Nutzern Mittel zur Verfügung, um den Schaden gering zu halten.
Response-Prüfung	Die Arbeit der Benutzer wird zu einem bestimmten Grad erhalten. (A)
Usability-Pattern(s)	[BJK01] 8. Recovering from Failure: Ein System könnte plötzlich, während ein Nutzer damit arbeitet, nicht mehr funktionieren. Nutzer sollten die Mittel zur Verfügung gestellt bekommen, um möglichst wenige Daten durch Systemfehler zu verlieren.
Interaktionskategorien	Behandlung von Fehlern und Hilfe
Usability-Ziel(e)	(+) Sicherheit: Der Einfluss von Fehlern oder Systemabstürzen auf die Benutzerdaten ist gering.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 87: Interaktionsszenario Wiederherstellung nach Betriebsausfall (Recovering From Failure)

Name	*Beschreibung*
Nummer	27
Kurzname	Vergessenes Passwort zurückbekommen (Retrieving Forgotten Passwords)
Quelle	Benutzer
Stimulus	haben ihr Passwort vergessen und möchten dennoch auf das System zugreifen.
Response	Um den Nutzern Zugriff zu garantieren, wird ein alternativer, sicherer Mechanismus verwendet.
Response-Prüfung	Berechtigte Benutzer erhalten Zugang zum System, auch wenn sie sich nicht mehr an ihr Passwort erinnern. (U)
Usability-Pattern(s)	[BJK01] 9. Retrieving Forgotten Passwords: Ein Nutzer könnte ein Passwort vergessen. Systeme sollten alternative, sichere Mechanismen zur Verfügung stellen, um Nutzern Zugang zu garantieren.
Interaktionskategorien	Behandlung von Fehlern und Hilfe
Usability-Ziel(e)	(-) Effizienz: Eine Alternative ist meist aufwendiger als die Passworteingabe. (+) Effektivität: Die Benutzer können, nachdem sie auf alternativem Weg Zugriff erlangten, mit den aktuellen Aufgaben fortfahren. (+) Sicherheit: Benutzer kann auf eigene Daten zugreifen. Sie sind nicht wegen des Passwortverlustes verloren.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 88: Interaktionsszenario Vergessene Passwörter zurückbekommen (Retrieving Forgotten Passwords

Name	*Beschreibung*
Nummer	28
Kurzname	Weiterverwenden von Informationen (Reusing Information)
Quelle	Benutzer
Stimulus	möchten einige Daten in einem anderen Teil des Systems nutzen.
Response	Das System bietet einen Mechanismus an, mit dem Daten kopiert und eingefügt werden können.
Response-Prüfung	Die kopierten Daten sind mit den Originaldaten identisch und können auf dieselbe Art und Weise verwendet werden. (A, U) Benutzer kopieren Daten und fügen sie an der/den von ihnen ausgewählten Stelle(n) ein. (U)
Usability-Pattern(s)	[BJK01] 11. Reusing Information: Ein Nutzer könnte Daten von einem Teil des Systems in einen anderen verlagern wollen. Nutzer sollten manuelle oder automatische Datentransportmöglichkeiten zwischen verschiedenen Teilen eines Systems zur Verfügung haben.
Interaktionskategorien	Austausch und Manipulation von Daten Kooperation (in einem System, mit anderen Systemen) Erfassung und Behandlung von Benutzereingaben
Usability-Ziel(e)	(+) Effizienz: Kein Zeitverlust durch wiederholte Dateneingabe.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 89: Interaktionsszenario Weiterverwenden von Informationen (Reusing Information)

Name	*Beschreibung*
Nummer	29
Kurzname	Varianten des Einfügens (Paste Variations)
Quelle	Benutzer
Stimulus	kopieren einige Daten und möchten sie an anderer Stelle wieder einfügen.
Response	Nutzer können das Format der eingefügten Daten wählen.
Response-Prüfung	Wenn die Daten eingefügt werden, behalten die Daten die von den Benutzern benötigten Eigenschaften. (U)
Usability-Pattern(s)	[Tido5] 87. Paste Variations: Stelle spezielle Einfügefunktionalitäten zusätzlich zu den Standardeinfügefunktionen bereit.
Interaktions-kategorien	Austausch und Manipulation von Daten Kooperation (in einem System, mit anderen Systemen)
Usability-Ziel(e)	(+) Effizienz: Kein Zeitverlust durch wiederholte Dateneingabe und Umformatierung.
Pot. SA-sensitivität	2*: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 90: Interaktionsszenario Varianten des Einfügens (Paste Variations)

Name	*Beschreibung*
Nummer	30
Kurzname	Multitasking (Supporting Multiple Activities)
Quelle	Benutzer
Stimulus	möchten zwischen mehreren, parallel bearbeiteten Aufgaben wechseln.
Response	Das System unterstützt das schnelle Vor- und Zurückwechseln zwischen Aufgaben.
Response-Prüfung	Das System bzw. die Anwendung ist nach dem Kontextwechsel stabil (A). Das Wechseln ist innerhalb einer bestimmten Zeitspanne abgeschlossen. (A) Benutzer unterbrechen ihre Arbeit an der aktuellen Aufgabe und wechseln zu einer anderen Aufgabe. (U) Benutzer wechseln zu einer Aufgabe, die unterbrochen war, und setzen ihre Arbeit daran fort. (U)
Usability-Pattern(s)	[BJK01] 15. Supporting Multiple Activities: Nutzer müssen oft an mehreren Aufgaben (mehr oder weniger) simultan arbeiten. Ein System oder dessen Anwendungen sollten dem Nutzer erlauben, zwischen diesen Aufgaben schnell hin und her zu wechseln.
Interaktionskategorien	Kooperation (in einem System, mit anderen Systemen) Navigieren, Browsen, Auswählen
Usability-Ziel(e)	(+) Effizienz: Kein Zeitverlust durch sequentielles Abarbeiten von Aufgaben, die mit Multitasking schneller bearbeitet werden können.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 91: Interaktionsszenario Multitasking (Supporting Multiple Activities)

Name	*Beschreibung*
Nummer	31
Kurzname	Benutzer-Tempo (User's Pace)
Quelle	Benutzer
Stimulus	möchten Interaktionen so schnell wie möglich ausführen, aber nicht zu schnell.
Response	Das System legt die angemessene Geschwindigkeit für Interaktionen mit den entsprechenden Nutzern fest (z.B. automatisches Scrolling) und ermöglicht es ihnen, die Geschwindigkeit anzupassen.
Response-Prüfung	Benutzer erledigen ihre Aufgaben während der Benutzung ohne Probleme wegen des vom System vorgegebenen Tempos. (U) Benutzer passen die Geschwindigkeit der Benutzer-System-Interaktionen an ein Level an, das ihnen angenehm ist. (U)
Usability-Pattern(s)	[BJK01] 18. Working at the User's Pace: Es könnte sein, dass sich ein System nicht dem Arbeitstempo eines Nutzers anpassen kann. Systeme sollten menschliche Bedürfnisse und Fähigkeiten berücksichtigen, wenn die einzelnen Phasen einer Interaktion durchschritten werden. Systeme sollten Nutzern erlauben, das (Interaktions-) Tempo anzupassen, wenn nötig.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben
Usability-Ziel(e)	(+) Nützlichkeit: Benutzergruppen, die langsamer oder schneller mit dem System interagieren wollen als voreingestellt, können dies tun.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 92: Interaktionsszenario Benutzer-Tempo (User's Pace)

Name	*Beschreibung*
Nummer	32
Kurzname	Fortschrittsanzeige (Progress Indication)
Quelle	Benutzer
Stimulus	möchten über den Fortschritt oder die verbleibende Zeit bis zum Ende eines Vorgangs informiert werden.
Response	Der Fortschritt des Vorgangs sowie die benötigte Zeit werden an ein Interface-Element fortwährend übermittelt und von diesem dargestellt.
Response-Prüfung	Die kalkulierte Dauer und die tatsächliche Dauer unterscheiden sich nur bis zu einem bestimmten Wert. (A) Benutzer können anhand einer grafischen Darstellung und/oder einer Zahl abschätzen, wie lang eine Aktion noch andauern wird. (U)
Usability-Pattern(s)	[Tid05] 49. Progress Indicator: Zeige dem Benutzer an, wie weit eine zeitaufwendige Aktion fortgeschritten ist. [BJK01] 19. Predicting Task Duration: Ein Nutzer sollte eine fundierte Entscheidung darüber treffen können, was er macht, wenn das System länger braucht, um einen Befehl auszuführen. Systeme sollten die angenommene Laufzeit für die Erfüllung einer Aufgabe anzeigen.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen Orientierung
Usability-Ziel(e)	(+) Nützlichkeit: In der Wartezeit können alternative Aufgaben durchgeführt werden. (+) Erlernbarkeit: Die Dauer einer Tätigkeit wird erlernt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 93: Interaktionsszenario Fortschrittsanzeige (Progress Indication)

Name	*Beschreibung*
Nummer	33
Kurzname	Umfassende Suche (Comprehensive Searching)
Quelle	Benutzer
Stimulus	suchen über mehrere Subsysteme hinweg nach Daten unterschiedlichen Typs.
Response	Umfangreich und konsistent werden anhand relevanter Kriterien Daten ermittelt und präsentiert.
Response-Prüfung	Alle suchbaren Arten von Medien werden auf dieselbe Art und Weise in ihrem Erscheinungsbild und der angeboten Funktionalität dargestellt. (A) Benutzer können erfolgreich anhand der Kriterien suchen, die sie als adäquat empfinden. (U) Benutzer finden den gewünschten Inhalt innerhalb einer bestimmten Zeitspanne. (U)
Usability-Pattern(s)	[BJK01] 20. Supporting Comprehensive Searching: Ein Nutzer möchte einige Dateien suchen oder Teile dieser Dateien nach verschiedenen Inhaltstypen durchsuchen. Systeme sollten Nutzern erlauben, Daten umfassend und konsistent zu suchen.
Interaktionskategorien	Informationsbeschaffung Selektion und Exploration von Daten Kooperation (in einem System, mit anderen Systemen)
Usability-Ziel(e)	(+) Effizienz: Gesuchte Informationen werden schnell ermittelt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 94: Interaktionsszenario Umfassende Suche (Comprehensive Searching)

Name	*Beschreibung*
Nummer	34
Kurzname	Vertrautes Aussehen und Verhalten (Familiar Appearance and/or Behaviour)
Quelle	Benutzer
Stimulus	möchten eine ihnen unbekannte Anwendung oder einen Teil davon nutzen.
Response	Das Aussehen und Verhalten der neuen Software entspricht der Software, die Nutzern schon bekannt ist.
Response-Prüfung	Das System agiert robust (siehe Szenario Alternativansichten (Alternative Views)). Benutzer beginnen sofort mit der Arbeit, da sie die neue oder geänderte Software verwenden können, ohne ein neues Bedienkonzept zu erlernen (U).
Usability-Pattern(s)	[BJK01] 13. Leveraging Human Knowledge: Ein Nutzer muss eine neue Anwendung mit einem bekannten Betriebssystem, eine neue Version einer bekannten Anwendung oder ein neues Produkt einer bekannten Produktlinie verwenden. Systemdesigner sollten versuchen, Upgrades zu entwickeln, die das Wissen der Nutzer über das vorhergehende System berücksichtigen, um ihnen somit zu ermöglichen, schnell und effizient mit dem neuen System umzugehen. [FGB03] 30. Emulation: Bilde das Aussehen und/oder das Verhalten eines anderen Systems nach.
Interaktionskategorien	Interagieren mit unbekannten Systemen Strukturieren und Anzeigen von Content, Informationen, Daten Navigieren, Browsen, Auswählen Anpassen durch Benutzer, für Benutzer, für Aufgaben
Usability-Ziel(e)	(+) Erlernbarkeit: Bekannte Interaktionen können genauso eingesetzt werden wie bisher.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 95: Interaktionsszenario Vertrautes Aussehen und Verhalten (Familiar Appearance and/or Behaviour)

Name	*Beschreibung*
Nummer	35
Kurzname	Wechsel zwischen neuem und altem User Interface (Switching between New and Traditional Interfaces)
Quelle	Benutzer
Stimulus	möchten eine ihnen unbekannte Anwendung oder einen Teil davon nutzen.
Response	Nutzer können entscheiden, ob sie das neue oder das alte, ihnen bekannte Design des User Interfaces verwenden.
Response-Prüfung	Wechsel zwischen User Interface Versionen erfolgen robust.
Usability-Pattern(s)	[BJK01] 13. Leveraging Human Knowledge: Ein Nutzer muss eine neue Anwendung mit einem bekannten Betriebssystem, eine neue Version einer bekannten Anwendung oder ein neues Produkt einer bekannten Produktlinie verwenden. Systemdesigner sollten versuchen, Upgrades zu entwickeln, die das Wissen der Nutzer über das vorhergehende System berücksichtigen, um ihnen somit zu ermöglichen, schnell und effizient mit dem neuen System umzugehen.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Interagieren mit unbekannten Systemen
Usability-Ziel(e)	(+) Nützlichkeit: Verschiedene Benutzergruppen können zwischen ihnen angenehmeren Varianten des User Interfaces wählen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 96: Interaktionsszenario Wechsel zwischen neuem und altem User Interface (Switching between New and Traditional Interfaces)

Name	*Beschreibung*
Nummer	36
Kurzname	Wortreiches User Interface (Verbose Interface)
Quelle	Benutzer
Stimulus	möchten eine ihnen unbekannte Anwendung oder einen Teil davon nutzen.
Response	Das Interface produziert detaillierte Ausgaben und Erklärungen einzelner Schritte und/oder User-Interface-Elemente.
Response-Prüfung	Die Angaben sind korrekt und aktuell. (A) Es treten insbesondere bei Anfängern sehr wenige Benutzerfehler auf. (U)
Usability-Pattern(s)	[BJK01] 22. Working in an Unfamiliar Context: Ein Benutzer muss seine Aufgabe in einer neuen Situation abarbeiten. Systeme sollten für Nutzer, die in einem unbekannten Kontext arbeiten, ein Anfänger-Interface zur Verfügung stellen und eine gute Anleitung bieten.
Interaktions-kategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Interagieren mit unbekannten Systemen
Usability-Ziel(e)	(+) Nützlichkeit: Verschiedene Benutzergruppen (Anfänger, Fortgeschrittene, Experten) erhalten Unterstützung. (+) Erlernbarkeit: Durch die Erläuterungen werden Fragen sofort beantwortet, wenn sie auftreten.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 97: Interaktionsszenario Wortreiches User Interface (Verbose Interface)

Name	*Beschreibung*
Nummer	37
Kurzname	Prüfen notwendiger Ressourcen (Verifying Resources)
Quelle	Benutzer
Stimulus	möchten einen Vorgang starten, der Ressourcen benötigt, die nicht immer vollständig verfügbar sind.
Response	Das System startet einen Vorgang nur dann, wenn es geprüft und bestätigt hat, dass alle erforderlichen Ressourcen verfügbar sind.
Response-Prüfung	Die Prüfung erfolgt in einer bestimmten Zeitspanne (A). Das Feedback hinsichtlich der Prüfung und dem Ergebnis der Prüfung erfolgt fortwährend (U) (z.B. durch Szenario Fortschrittsanzeige (Progress Indication) und Szenario 1 System-Feedback).
Usability-Pattern(s)	[BJK01] 23. Verifying Resources: Eine Anwendung könnte daran scheitern, dass nicht genügend Ressourcen für einen Arbeitsablauf vorhanden sind. Diese Fehlfunktion kann zu unerwarteten Fehlern während des Programmablaufs führen. Anwendungen sollten verifizieren, dass alle notwendigen Ressourcen verfügbar sind, bevor mit einem Ablauf begonnen wird.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effektivität: Das System arbeitet zuverlässig. Komplexe und zeitaufwendige Interaktionen werden nur durchgeführt, wenn die Ressourcen vorhanden sind. (+) Sicherheit: Die Benutzerdaten werden durch diese Maßnahme zur Fehlervermeidung beschützt.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 98: Interaktionsszenario Prüfen notwendiger Ressourcen (Verifying Resources)

Name	*Beschreibung*
Nummer	38
Kurzname	Gleiche Bedienung verschiedener Ansichten (Operating Consistently Across Views)
Quelle	Benutzer
Stimulus	arbeiten mit einem Datensatz und verwenden verschiedene Ansichten des Datensatzes.
Response	Das System stellt in den verschiedenen Ansichten immer alle möglichen Funktionen für die Daten zur Verfügung, unabhängig von der Darstellung der Daten.
Response-Prüfung	Die Funktionen sind immer komplett verfügbar. (A) Die Benutzer finden die jeweiligen Funktionen in einer bestimmten Zeitspanne (U).
Usability-Pattern(s)	[BJK01] 24. Operating Consistently Across Views: Ein Nutzer könnte durch den abweichenden Funktionsumfang in verschiedenen Ansichten auf dieselben Daten verwirrt werden. Systeme sollten unabhängig von der Darstelung der Daten immer die gleichen Befehle zur Verfügung stellen. Diese sollten auf dem Typ und dem Inhalt der Daten eines Nutzers basieren und nicht auf der gegenwärtigen Ansicht der Daten.
Interaktionskategorien	Ausführen, Wiederholen und Zurücknehmen von Befehlen Austausch und Manipulation von Daten
Usability-Ziel(e)	(+) Effizienz: Wenn immer das gleiche Set an Interaktionen möglich ist, muss nur einmal danach gesucht werden. (+) Erlernbarkeit: Wenn eine Ansicht bekannt ist, können folgende Ansichten sofort verwendet werden. (+) Einprägsamkeit: Wenn immer das gleiche Set an Interaktionen möglich ist, prägt sich dies gut ein (Routine).
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 99: Interaktionsszenario Gleiche Bedienung verschiedener Ansichten (Operating Consistently Across Views)

Name	*Beschreibung*
Nummer	39
Kurzname	Ansichten verfügbar machen (Making views accessible, war: Consistent Set of Views)
Quelle	Benutzer
Stimulus	arbeiten mit einem Datensatz und verwenden verschiedene Ansichten des Datensatzes.
Response	Die verfügbaren Ansichten sind in jedem angebotenen Betriebsmodus gleich.
Response-Prüfung	Der Kontextwechsel verläuft zügig und sicher. (A)
Usability-Pattern(s)	[BJK01] 25. Making Views Accessible: Nutzer möchten oft Daten aus verschiedenen Perspektiven betrachten (z.B. Inhaltsverzeichnis und Text). Sind bestimmte Ansichten in einem Betriebsmodus nicht verfügbar, oder ist ein Wechsel zwischen verschiedenen Ansichten umständlich, erschwert dies dem Nutzer seine Arbeit mit der Anwendung.
Interaktionskategorien	Selektion und Exploration von Daten
Usability-Ziel(e)	(+) Effizienz: Wenn Nutzer häufig zwischen den Ansichten wechseln können, erfahren sie schnell viel über die Daten und können so zügig arbeiten. (+) Erlernbarkeit: Wenn die angebotenen Sichten aus einer Sicht bekannt sind, können folgende Ansichten sofort verwendet werden. (+) Einprägsamkeit: Wenn immer das gleiche Set an möglichen Ansichten, prägt sich dies gut ein (Routine).
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 100: Interaktionsszenario Ansichten verfügbar machen (Making views accessible, war: Consistent Set of Views)

Name	*Beschreibung*
Nummer	40
Kurzname	Strukturieren und Verstecken von Content (Structuring and Hiding Content)
Quelle	System
Stimulus	hat mehr Inhalte zur Verfügung als Platz auf dem Bildschirm/einem Fenster oder als wahrnehmbar für den Nutzer.
Response	Das System strukturiert Inhalte visuell und versteckt dabei Inhalte, die gegenwärtig nicht genutzt werden.
Response-Prüfung	Benutzer finden die benötigten Informationen schnell. (U)
Usability-Pattern(s)	[Tido5] 35. Card Stack: Platziere inhaltlich voneinander verschiedene Abschnitte auf separaten Kartendecks und erstelle einen Stapel, so dass immer nur eine Karte sichtbar ist. Verwende Tabulatoren oder andere grafische Elemente/Shortcuts, damit die Nutzer die einzelnen Karten auswählen und einsehen können. [Tido5] 36. Closable Panels: Biete Inhaltsabschnitte auf einzelnen Panels (beispielsweise Terminals, Bedienfeld, Browser-Tabs) an, die unabhängig voneinander geöffnet und geschlossen werden können [Tido5] 38. Right/Left Alignment: Richte bei einem zweispaltigen Formular (oder einer Tabelle) die Elemente der linken Seite rechtsbündig und jene der rechten Seite linksbündig aus. [Tido5] 74. Dropdown Chooser: Erweitere das Konzept eines Menüs durch den Einsatz eines Dropdown- oder Pop-up-Panels, um komplexere Auswahlen zu ermöglichen.
Interaktionskategorien	Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effizienz: Informationen sind zwar versteckt, aber schon abrufbereit.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 101: Interaktionsszenario Strukturieren und Verstecken von Content (Structuring and Hiding Content)

Name	*Beschreibung*
Nummer	41
Kurzname	Bewegbare Panels (Movable Panels, war: Amendable Screen Layout)
Quelle	System
Stimulus	hat mehr Inhalte zur Verfügung als Platz auf dem Bildschirm/einem Fenster oder als wahrnehmbar für den Nutzer.
Response	Nutzer können die einzelnen Panels auf dem Bildschirm verschieben und damit das Layout ändern.
Response-Prüfung	Das Layout wird automatisch anhand von Regeln bzw. im Rahmen von Grenzen gesetzt, damit das Layout in jeder Variante ästhetisch gestaltet ist. (A)
Usability-Pattern(s)	[Tido5] 37. Movable Panels: Platziere verschiedene Werkzeuge oder inhaltliche Abschnitte in separate Panels, die von Nutzern bewegt werden können, um ein kundenspezifisches Layout zu erstellen.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Nützlichkeit: Benutzer können das System so anpassen, dass es für sie den besten Nutzen bietet.
Pot. SA-sensitivität	2*: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 102: Interaktionsszenario Bewegbare Panels (Movable Panels, war: Amendable Screen Layout)

Name	*Beschreibung*
Nummer	42
Kurzname	Dynamische Abfrage (Dynamic Query)
Quelle	Benutzer
Stimulus	möchten schnell einen Datensatz durchsuchen, der schon vorhanden, aber nur zum Teil auf dem Bildschirm sichtbar ist.
Response	Die gewünschten Daten werden automatisch zusammengestellt und in der Ansicht zur Verfügung gestellt.
Response-Prüfung	Die Daten werden in einer bestimmten Zeitspanne zur Verfügung gestellt. (A) Notwendige Ressourcen sind abgesichert (z.B. Speicherplatz für Caching). (A) Nutzer navigieren mit dem geringst möglichen Aufwand sowie der geringst möglichen Verzögerung. (U)
Usability-Pattern(s)	[BJK01] 16. Navigating Within a Single View: Ein Nutzer könnte von Daten, die auf dem Bildschirm zu sehen sind, zu nicht sichtbaren Daten navigieren wollen (z.B. untere Einträge in einer Liste). Systemdesigner sollten es ermöglichen, dass Nutzer innerhalb einer Ansicht navigieren können und dass Wartezeiten minimal sind. [Tid05] 56. Dynamic Queries: Ermögliche es, den Datensatz sofort und interaktiv zu filtern. Setze dafür Standard-Bedienelemente ein (Auswahlfelder, Schieberegler), damit die Benutzer auf einfache Weise festlegen können, welche Teile des Datensatzes angezeigt werden sollen. Sobald die Benutzer mit ihren Änderungen fertig sind, erscheinen die Ergebnisse in der Anzeige. [Tid05] 61. Jump to Item: Wenn der Nutzer den Namen eines Elementes eintippt, springe direkt zu diesem Eintrag und selektiere ihn.
Interaktionskategorien	Selektion und Exploration von Daten Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effizienz: Weniger Interaktionen notwendig bzw. weniger zeitaufwendige Interaktionen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 103: Interaktionsszenario Dynamische Abfrage (Dynamic Query)

Name	*Beschreibung*
Nummer	43
Kurzname	Fehlermeldungen (Error Message)
Quelle	System
Stimulus	gibt eine Fehlermeldung aus.
Response	Nutzer erhalten kurze spezifische Informationen, um das Problem zu lösen. Die Fehler werden protokolliert und enthalten kodierte Informationen über die Fehlerkonditionen für die Entwickler.
Response-Prüfung	Das System ist trotz Fehlern stabil und robust. (A) Die Fehler der Benutzer haben geringe Auswirkungen. (U) Die Fehlerrate der Benutzer sinkt. (U)
Usability-Pattern(s)	[Lit10] 10. Error Messages: Wenn möglich, kategorisiere Fehler: ob der Benutzer das Problem lösen kann, er es später probieren soll oder ob ein nichtlösbares Problem aufgetreten ist. Natürlich können viele für den Benutzer momentan unlösbare Probleme von Entwicklern gelöst werden. Die besten Befehle für Fehlermeldungen sind „Erneut versuchen“, „Eingabe überarbeiten“, „Abbrechen“, „Daten sichern“ (Aufgabe abbrechen ohne Arbeitsergebnisse zu verlieren). Selten eignet sich „OK“.
Interaktions-kategorien	Behandlung von Fehlern und Hilfe Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effektivität: Die Benutzer können nach der Fehlerbehebung (falls möglich) ihre Aufgabe fertig stellen. Durch das Sichern von Daten werden Benutzerdaten (Daten, mit denen sie gearbeitet haben) erhalten. (+) Sicherheit: Fehler werden behandelt und behoben. Durch das Sichern von Daten werden Benutzerdaten (Daten, mit denen sie gearbeitet haben) erhalten. (+) Erlernbarkeit: Die Benutzer erfahren, wie sie zukünftig Fehler vermeiden können.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 104: Interaktionsszenario Fehlermeldungen (Error Message)

Name	*Beschreibung*
Nummer	44
Kurzname	Wizard
Quelle	Benutzer
Stimulus	wollen eine Aufgabe fertigstellen, die entweder neu für sie ist oder kaum erfüllt wurde und die aus einer Reihe von einzelnen Entscheidungen besteht.
Response	Die Aufgabe wird in Teilaufgaben zerteilt, die Nutzer werden angeleitet und erledigen sie Schritt für Schritt.
Response-Prüfung	Die Aufgabe wird durch das System in einer bestimmten Zeitspanne sicher und korrekt ausgeführt. Es wird Feedback über den Fortschritt gegeben (A). Die Fehlerrate der Benutzer ist sehr gering. (U)
Usability-Pattern(s)	[Tid05] 17. Wizard: Führe den Nutzer Schritt für Schritt durch das Interface, um Aufgaben in einer vorgeschriebenen Weise zu erfüllen. [FGB03] 34. Wizard: Der Nutzer möchte ein Ziel erreichen und dafür müssen mehrere Entscheidungen getroffen werden. Es kann sein, dass der Nutzer dies (noch) nicht weiß. Führe den Nutzer Schritt für Schritt durch die gesamte Aufgabe. Zeige an, welche Schritte es gibt und welche bereits ausgeführt wurden.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Erfassung und Behandlung von Benutzereingaben Interagieren mit unbekannten Systemen
Usability-Ziel(e)	(+) Effizienz: Anfänger sparen Zeit für die Exploration des User Interfaces. (-) Effizienz: Experten werden mit Anleitung aufgehalten. (+) Sicherheit: Durch Anleitung wird eine bestimmte Reihenfolge eingehalten, werden bestimmte Daten abgerufen. Fehlerhafte Angaben können rechtzeitig behoben werden. (+) Nützlichkeit: Anfänger erhalten gesonderte Unterstützung. (+) Erlernbarkeit: Anleitung ermöglicht schnelles Lernen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 105: Interaktionsszenario Wizard

Name	*Beschreibung*
Nummer	45
Kurzname	Allmähliche Offenlegung (Responsive Disclosure/Enabling)
Quelle	Benutzer
Stimulus	möchten eine komplexe Aufgabe abarbeiten, die aus einer Reihe von Unteraufgaben besteht.
Response	In einer bestimmten Reihenfolge werden zu bewältigende Aufgaben gestellt und/oder einzelne Funktionen freigeschaltet.
Response-Prüfung	Die Anzeige bzw. Freischaltung der nächsten Aufgaben bzw. Funktionen erfolgt in einer bestimmten Zeitspanne. (A)
Usability-Pattern(s)	[Tid05] 41. Responsive Disclosure: Leite den Nutzer durch eine Folge von Schritten an, wobei ausgehend von einem minimalen User Interface schrittweise mehr vom User Interface gezeigt wird. [Tid05] 42. Responsive Enabling: Leite den Nutzer durch eine Folge von Schritten an, wobei ausgehend von einem User Interface mit minimaler Funktionalität schrittweise mehr und mehr Funktionen freigegeben und aktiviert werden.
Interaktionskategorien	Erfassung und Behandlung von Benutzereingaben Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Sicherheit: Es ist immer klar, was zu tun ist und was sinnvoll ist. Funktionen, die zu Fehlern führen könnten, sind nicht freigeschaltet. (+) Erlernbarkeit: Der Benutzer erhält erst nach erledigten Arbeitsschritten Einblick in weitere mögliche Interaktionen. Es ist immer ohne Ablenkung klar, was jetzt in der Situation getan werden muss.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 106: Interaktionsszenario Allmähliche Offenlegung (Responsive Disclosure/Enabling)

Name	*Beschreibung*
Nummer	46
Kurzname	Extras auf Wunsch (Extras on Demand)
Quelle	Benutzer
Stimulus	greifen häufig auf einen bestimmten Teildatensatz des dargebotenen Inhaltes zu, während die restlichen Daten kaum genutzt werden.
Response	Nutzer können direkt auf den wichtigsten Teil des Inhalts zugreifen. Der Rest des Inhalts ist über einen zusätzlichen Schritt erreichbar.
Response-Prüfung	Alle zusätzlichen Daten sind innerhalb einer bestimmten Zeitspanne zugänglich (A). Die Benutzer sind mit der Bestimmung, was ein „wichtiger Inhalt“ ist, einverstanden. (U)
Usability-Pattern(s)	[Tid05] 18. Extras on Demand: Zeige den wichtigsten Inhalt zuerst und verstecke den Rest. Lasse den Nutzer mit einer einzelnen, einfachen Geste dahin gelangen. Show the most important content up front, but hide the rest. Let the user reach it via a single, simple gesture.
Interaktionskategorien	Navigieren, Browsen, Auswählen Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effizienz: Die wesentlichen Informationen sind sofort sichtbar.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 107: Interaktionsszenario Extras auf Wunsch (Extras on Demand)

Name	*Beschreibung*
Nummer	47
Kurzname	Mehrstufige Hilfe (Multi-Level Help)
Quelle	Benutzer
Stimulus	brauchen Hilfe im Umgang mit dem System.
Response	Das System stellt kontextabhängig die geeignete Art Hilfe zur Verfügung.
Response-Prüfung	Die Hilfe ist jederzeit in einer bestimmten Zeitspanne verfügbar (A). Die Hilfe bietet Information und/oder korrekt Anleitung (U).
Usability-Pattern(s)	[Tid05] 20. Multi-Level Help: Verwende eine Mischung aus verschiedenen Hilfetechniken (technisch einfach und technisch komplexer zu realisieren), um Nutzer mit unterschiedlichen Bedürfnissen zu unterstützen. [BJK01] 10. Providing Good Help: Ein Nutzer braucht Hilfe. Hilfsprozeduren sollten kontextabhängig und ausreichend vollständig sein, um Nutzern zu helfen, ein Problem zu lösen. [FGB03] 31. Context Sensitive Help: Beobachte, was der Nutzer gegenwärtig tut und stelle Dokumentation zur Verfügung, die relevant für die Erfüllung der Aufgabe ist.
Interaktions-kategorien	Behandlung von Fehlern und Hilfe
Usability-Ziel(e)	(+) Effizienz: Hilfe vor Ort ist schnell verfügbar und beantwortet Fragen, die sich aus der aktuellen Situation ergeben, sofort. Es kommt nicht zu Verzögerungen durch die Suche nach Antworten. (+) Sicherheit: Durch die Anleitung werden Benutzerfehler verhindert, die auf Unkenntnis beruhen.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 108: Interaktionsszenario Mehrstufige Hilfe (Multi-Level Help)

Name	*Beschreibung*
Nummer	48
Kurzname	Sortierbare Tabelle (Sortable Table, war: Retrieving Information in a Table)
Quelle	Benutzer
Stimulus	möchten einen Datensatz in einer Tabelle schnell finden.
Response	Die Tabelle ist nach jeder Spalte sortierbar.
Response-Prüfung	Die Sortierung erfolgt in einer bestimmten Zeitspanne (A).
Usability-Pattern(s)	[Tid05] 60. Sortable Table: Zeige die Daten in einer Tabelle an und lasse Benutzer die Tabellenreihen entsprechend der Spaltenwerte sortieren.
Interaktions-kategorien	Informationsbeschaffung Strukturieren und Anzeigen von Content, Informationen, Daten
Usability-Ziel(e)	(+) Effizienz: Das Sortieren ermöglicht ein schnelles Auffinden eines gesuchten Eintrages.
Pot. SA-sensitivität	1*: Einfache Modifikationen innerhalb einer (mehrerer) Komponente(n). Neue Einschätzung durch einen Experten.

Tabelle 109: Interaktionsszenario Sortierbare Tabelle (Sortable Table, war: Retrieving Information in a Table)

Name	*Beschreibung*
Nummer	49
Kurzname	Kontextbewusste Interaktion (Context-aware Interaction)
Quelle	Benutzer
Stimulus	möchten eine kontextabhängige Aufgabe erfüllen.
Response	Das System gleicht die Auswahl der zur Verfügung gestellten Werkzeuge/Aktionen beständig mit dem sich verändernden Kontext ab.
Response-Prüfung	Überwachung der Kontextänderungen erfolgt durchgehend. (A) Die Reaktion auf Kontextänderungen erfolgt innerhalb einer bestimmten Zeitspanne. (A)
Usability-Pattern(s)	[FGB03] 29. Workflow Model: Stelle verschiedenen Nutzern nur die Werkzeuge oder Aktionen zur Verfügung, die für konkrete Aufgaben bezüglich eines Ausschnittes von Daten nötig sind.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Ausführen, Wiederholen und Zurücknehmen von Befehlen
Usability-Ziel(e)	(+) Effizienz: Die Benutzer müssen sich nicht mit dem Überwachen von Daten beschäftigen, sondern können sich effizient auf andere Tätigkeiten konzentrieren. (+) Effektivität: Die Aufgabe wird dann gestartet, wenn sie abgearbeitet werden muss.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 110: Interaktionsszenario Kontextbewusste Interaktion (Context-aware Interaction)

Name	*Beschreibung*
Nummer	50
Kurzname	Neustart einer Anwendung (Re-entering an Application)
Quelle	Benutzer
Stimulus	möchten eine Anwendung nach einem Absturz oder Systemausfall erneut betreten.
Response	Die Anwendung startet korrekt und präsentiert die Benutzerdaten in einem Zustand, bevor ein Fehler aufgetreten ist.
Response-Prüfung	Der Zustand ist stabil (A). Die Benutzer starten sofort mit der Arbeit (U).
Usability-Pattern(s)	[BJK01] 8. Recovering from Failure: Ein System könnte plötzlich, während Nutzer damit arbeiten, nicht mehr funktionieren. Nutzer sollten Mittel zur Verfügung gestellt bekommen, um möglichst wenige Daten durch Systemfehler zu verlieren.
Interaktionskategorien	Anpassen durch Benutzer, für Benutzer, für Aufgaben Orientierung Behandlung von Fehlern und Hilfe
Usability-Ziel(e),	(+) Effizienz: Die Benutzer können sofort an der Stelle weiter arbeiten, an der sie aufgehört haben oder aufhören mussten. (+) Sicherheit: Die Daten der Benutzer werden gesichert.
Pot. SA-sensitivität	2***: Komplexe Modifikationen durch das Ergänzen von Komponente(n). Mehrere Beispiele und Dokumente (siehe Usability-Patterns) bestätigen, dass die Einschätzung meistens zutrifft.

Tabelle 111: Interaktionsszenario Neustart einer Anwendung (Re-entering an Application)

In Qualitätsmodellen wird Softwarequalität für Anwendungen spezifiziert und damit leichter prüfbar [Int00]. Qualitätsmodelle unterteilen Qualität in Faktoren, die wiederum weitere Begriffe umfassen. Abhängigkeiten zwischen Faktoren werden nicht betrachtet. Die Perspektiven, aus denen die Qualitätsmodelle Softwarequalität betrachten, die Bezeichnung der Faktoren und die Anzahl der spezifizierten Begriffe sind unterschiedlich. Die Modelle sind sich inhaltlich sehr ähnlich, obwohl sie aus verschiedenen Perspektiven erstellt wurden. Alle drei klassifizieren Usability als Softwarequalitätsmerkmal. Tabelle 112 stellt diese drei Qualitätsmodelle gegenüber, indem inhaltlich ähnliche Faktoren in einer Zeile zusammengefasst werden. Daraus entsteht eine zusammenfassende Liste von 14 Qualitätsmerkmalen. Dabei ist zu beachten, dass manche Faktoren in den Qualitätsmodellen zwei oder mehr Qualitätsmerkmale beeinflussen: beispielsweise ist Regelkonformität im Modell der ISO [Int00] sowohl ein Faktor für Portierbarkeit als auch für Verlässlichkeit, Effizienz und Usability. Das weißt auf Austauschbeziehungen (oder „Wechselwirkungen“) zwischen verschiedenen Qualitätsmerkmalen hin. Diese sind beim Entwurf einer Softwarearchitektur zu beachten und zu balancieren.

Qualitätsmerkmal	*McCall et al.* (Unternehmenssicht)	*Boehm et al.* (Entwicklersicht)	*ISO 9126* (Benutzersicht)
Korrektheit	Korrektheit	Korrektheit	Wartbarkeit
Funktionalität	–	–	Funktionalität
Verlässlichkeit	Verlässlichkeit	Verlässlichkeit	Verlässlichkeit
Effizienz	Effizienz	Effizienz	Effizienz
Integrität	Integrität	Integrität	–
Usability	Usability	Usability	Usability
Wartbarkeit	Wartbarkeit	–	Wartbarkeit
Modifizierbarkeit	–	Modifizierbarkeit	Wartbarkeit
Testbarkeit	Testbarkeit	Testbarkeit	Wartbarkeit
Flexibilität	Flexibilität	Modifizierbarkeit	Flexibilität
Portierbarkeit	Portierbarkeit	Portierbarkeit	Portierbarkeit
Wiederverwendbarkeit	Wiederverwendbarkeit	Portierbarkeit	Portierbarkeit
Interoperabilität	Interoperabilität	–	–
Verständlichkeit	–	Verständlichkeit	–

Tabelle 112: Qualitätsmodelle im Überblick (inhaltlich ähnliche Faktoren in einer Zeile)

[ABC+96] Gregory Abowd, Len Bass, Paul Clements, Rick Kazman, Linda Northrop, and Amy Zamerski, *Recommended best industrial practice for software architecture evaluation*, Tech. report, (CMU/SEI-96-TR-025), Software Engineering Institute, Carnegie Mellon University, 1996.

[AIS78] Christopher Alexander, Sara Ishikawa, and Murray Silverstein, *A pattern language: Towns, buildings, constructions*, Construction Oxford University Press, 1978.

[ALMN99] David E. Avison, Francis Lau, Michael D. Myers, and Peter Axel Nielsen, *Action research*, Communications of the ACM **42** (1999), no. 1, 94–97.

[Ant05] Peter Antoniac, *A taxonomy of mobility: Some implications and requirements for mobile information appliances*, Journal of Control Engineering and Applied Informatics **7** (2005), no. 5, 3–10.

[AP99] Josef Altmann und Gustav Pomberger, *Kooperative Softwareentwicklung: Konzepte, Modell und Werkzeuge*, 4. Int. Tagung Wirtschaftsinformatik 1999 (Markus Nüttgens A.-W. Scheer, ed.), Heidelberg: Physica Verlag, 1999.

[AW99] Alberto Avritzer and Elaine J. Weyuker, *Metrics to assess the likelihood of project success based on architecture reviews*, Empirical Software Engineering Journal **4** (1999), no. 3, 197–213.

[AZ05] Paris Avgeriou and Uwe Zdun, *Architectural patterns revisited – a pattern language*, Proc. of the 10th European Conference on Pattern Languages of Programs (EuroPLoP 2005), 2005, pp. 1–39.

[AZR11] E. Anjos and M. Zenha-Rela, *A framework for classifying and comparing software architecture tools for quality evaluation*, Proc. of the International Conference on Computational Science and its Applications ICCSA, Springer Berlin/Heidelberg, 2011, pp. 270–282.

[Bal07] Barbara Ballard, *Designing the mobile user experience*, Wiley & Sons, April 2007.

[BB98] Per Olof Bengtsson and Jan Bosch, *Scenario based software architecture reengineering*, Proc. of the International Conference of Software Reuse, 1998, pp. 308–317.

[BB99] ———, *Architecture level prediction of software maintenance*, Proc. of the 3rd European Conference on Software Maintenance and Reengineering, 1999.

[BB03] Robert Bogdan and Sara Knopp Biklen, *Qualitative research for education: An introduction to theories and methods*, Pearson Education: New York, 2003.

[BBC+03] Nigel Bevan, Carol Barnum, Gilbert Cockton, Jakob Nielsen, Jared Spool, and Dennis Wixon, *The „magic number 5": Is it enough for web testing?*, Proc. of the CHI '03: CHI '03 extended abstracts on human factors in computing systems (New York, NY, USA), ACM, 2003, pp. 698–699.

[BBK+81] Barry Boehm, J. R. Brown, H. Kaspar, M. Lipow, G. J. Macleod, and M. J. Merrit, *Characteristics of software quality*, 1981.

[BC89] Kent Beck and Ward Cunningham, *A laboratory for teaching object-oriented thinking*, Proc. of OOPSLA'89, 1989.

[BCK03] Len Bass, Paul Clements, and Rick Kazman, *Software architecture in practice, 2nd edition.*, SEI Series in Software Engineering, Addison-Wesley, 2003.

[BEL+03] Mario R. Barbacci, Robert Ellison, Anthony J. Lattanze, Judith A. Stafford, Charles B. Weinstock, and William G. Wood, *Quality attribute workshops (QAWS), third edition*, Tech. report, Carnegie Mellon University, Software Engineering Institute, Pittsburgh PA, 2003.

[Ber09] Bruce Berg, *Qualitative research methods for social sciences*, 7th ed., Pearson New York, 2009.

[Bev01] Nigel Bevan, *International standards for HCI and usability*, International Journal of Human Computer Studies **55** (2001), no. 4, 533–552.

[BFJK99] John K. Bergej, Matthew J. Fisher, Lawrence G. Jones, and Rick Kazman, *Software architecture evaluation with ATAMSM in the DoD system acquisition context*, Tech. report, Software Engineering Institute, Carnegie Mellon University, 1999.

[BG04] Muhammad Ali Babar and Ian Gorton, *Comparison of scenario-based software architecture evaluation methods*, Proc. of the 11th Asia-Pacific Software Engineering Conference (APSEC'04), 2004, BG04, pp. 600–607.

[BG07] Bettina Biel and Volker Gruhn, *Content adaptation*, Proc. of the VikingPLoP, 2007.

[BG09] ———, *Towards a method for analyzing architectural support levels of usability*, Proc. of the Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture (WICSA/ECSA 2009), Cambridge, 2009, pp. 273–276.

[BG10a] ———, *Analyzing the architectural support of usability*, Proc. of the 36th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2010), 2010, pp. 20–27.

[BG10b] ———, *Usability-improving mobile application development patterns*, Proc. of the 15th European Conference on Pattern Languages of Programs (EuroPLoP 2010), 2010.

[BGG08] Bettina Biel, Thomas Grill, and Volker Gruhn, *Patterns of trust*, Workshop Trustworthy Ubiquitous Computing, Proc. of the Mobile Multimedia (MoMM 2008), ACM, 2008, pp. 391–396.

[BGG10] ———, *Exploring the benefits of the combination of a software architecture analysis and a usability evaluation of a mobile application*, Interplay between Usability Evaluation and Software Development, Journal of Systems and Software **83** (2010), no. 11, 2031 – 2044.

[BJK01] Len Bass, Bonnie E. John, and Jesse Kates, *Achieving usability through software architecture*, Tech. report, Software Engineering Institute, Carnegie Mellon University, 2001.

[BLBV04] Per Olof Bengtsson, Nico Lassing, Jan Bosch, and Hans van Vliet, *Architecture-level modifiability analysis*, Journal of Systems and Software **69** (2004), 129–147.

[BM85] David Brinberg and Joseph E. McGrath, *Validity and the research process*, SAGE Publications, 1985.

[BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad, and Michael Stal, *Pattern-oriented software architecture, vol. 1: A system of patterns*, vol. 1, John Wiley & Sons, 1996.

[Bos00] Jan Bosch, *Design and use of software architectures: Adopting and evolving a productline approach*, Addison-Wesley, 2000.

[Bro90] D. B. Bromley, *Academic contributions to psychological counselling: I. a philosophy of science for the study of individual cases*, Counselling Psychology Quarterly **3** (1990), no. 3, 299–307.

[BRSS00] Frank Buschmann, Hans Rohnert, Peter Sommerlad, and Michael Stal, *Pattern-oriented software architecture, vol.2: Patterns for concurrent and networked objects*, vol. 2, John Wiley & Sons, 2000.

[BSG11] Bettina Biel, Stefan Seitz, and Volker Gruhn, *Towards pattern-based generic interaction scenarios*, Workshop Frontiers in Ambient and Mobile Systems, Proc. of the 8th Mobile Web Information Systems MobiWIS, 2011.

[BWHW05] Christian Braun, Felix Wortmann, Martin Hafner, and Robert Winter, *Method construction - a core approach to organizational engineering*, Proc. of the SAC'05, March 13-17, 2005, Santa Fe, New Mexico, USA, 2005, pp. 1295–1299.

[BZJ04] Muhammad Ali Babar, Liming Zhu, and Ross Jeffery, *A framework for classifying and comparing software architecture evaluation methods*, Proc. of the 5th Australian Software Engineering Conference, 2004, pp. 309–319.

[CBB+02] Paul Clements, Felix Bachmann, Len Bass, David Garlan, James Ivers, and Reed Little, *Documenting software architectures: Views and beyond*, Addison-Wesley Longman, Amsterdam, 2002.

[CHH05] Christine Choppy, Denis Hatebur, and Maritta Heisel, *Architectural patterns for problem frames*, IEE Proc. - Software, Special Issue on Relating Software Requirements and Architectures **152** (2005), no. 4, 196–208.

[CI06] Angela Chang and Hiroshi Ishii, *Sensorial interfaces*, DIS '06: Proc. of the 6th ACM conference on Designing Interactive systems (New York, NY, USA), ACM Press, 2006, pp. 50–59.

[CK06] Constantinos K. Coursaris and Dan J. Kim, *A qualitative review of empirical mobile usability studies*, Proc. of the 12th Americas Conference on Information Systems, Acapulco, 2006.

[CK07] ———, *A research agenda for mobile usability*, Proc. of the Computer-Human-Interaction (CHI 2007), 2007.

[CKK02] Paul Clements, Rick Kazman, and Mark Klein, *Evaluating software architectures: Methods and case studies*, SEI Series in Software Engineering, Addison-Wesley Longman, Amsterdam, 2002.

[CL99] Larry L. Constantine and Lucy A. D. Lockwood, *Software for use: A practical guide to the models and methods of usage-centred design*, Addison-Wesley, New York, NY, 1999.

[CM04] Scott Carter and Jennifer Mankoff, *Challenges for ubicomp evaluation*, Tech. Report UCB/CSD-04-1331, EECS Department, University of California, Berkeley, Jun 2004.

[CN08] Céline Coutrix and Laurence Nigay, *Balancing physical and digital properties in mixed objects*, AVI '08: Proc. of the working conference on Advanced visual interfaces (New York, NY, USA), ACM, 2008, pp. 305–308.

[Coc01] Alistair Cockburn, *Writing effective use cases*, Addison-Wesley, 2001.

[Coo95] Alan Cooper, *About face: The essentials of user interface design*, John Wiley & Sons, 1995.

[CRC07] Alan Cooper, Robert M. Reimann, and David Cronin, *About face 3.0: The essentials of interaction design*, 3rd edition ed., Wiley, May 2007.

[Deu11] Deutsches Institut für Normung, *Ergonomie der Mensch-System-Interaktion - Teil 210: Prozess zur Gestaltung gebrauchstauglicher interaktiver Systeme (ISO 9241-210:2010)*, Beuth, 2011.

[DFAB03] Alan Dix, Janet Finlay, Gregory D. Abowd, and Russell Beale, *Human-computer interaction*, 3rd ed., Prentice Hall, 2003.

[DMK04] Robert M. Davison, Maris G. Martinsons, and Ned Knock, *Principles of canonical action research*, Info Systems **14** (2004), 65–86.

[DN02] Liliana Dobrica and Eila Niemelä, *A survey on software architecture analysis methods*, IEEE Transactions on Software Engineering (2002), pp. 638–653.

[ESSD07] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian, *Selecting empirical methods for software engineering research*, Guide to Advanced Empirical Software Engineering (Forrest Shull, Janice Singer, and Dag I. K. Sjøberg, eds.), Springer, 2007.

[FB03] Eelke Folmer and Jan Bosch, *Usability patterns in software architecture*, Proc. of the 10th Int. Conf. on Human-Computer Interaction (HCII2003) Volume I., 2003.

[FB05] _____, *Analyzing software architectures for usability*, Proc. of the Euromicro Conference on Software Engineering, 2005.

[FGB03] Eelke Folmer, Jilles Van Gurp, and Jan Bosch, *A framework for capturing the relationship between usability and software architecture*, Software Process: Improvement and Practice (2003), 67–87.

[FJ06] Xavier Ferré and Natalia Juristo, *How to integrate usability into the software development process, ICSE tutorial*, 2006.

[FJM05] Xavier Ferré, Natalia Juristo, and Ana Moreno, *Framework for integrating usability practices into the software process*, PROFES 2005 (F. Bomarius and S. Komi-Sirviö, eds.), vol. LNCS 3547, Springer-Verlag Berlin Heidelberg, 2005, pp. 202–215.

[Fol05] Eelke Folmer, *Software architecture analysis of usability*, Ph.D. thesis, Rijksuniversiteit Groningen, 2005.

[FvGB04] Eelke Folmer, Jilles van Gurp, and Jan Bosch, *Software architecture analysis of usability*, Proc. of the 9th IFIP Working Conference on Engineering for Human Computer Interaction (Wiley, ed.), 2004, pp. 321–339.

[FvWBo6] Eelke Folmer, Martijn van Welie, and Jan Bosch, *Bridging patterns: An approach to bridge gaps between SE and HCI*, Information and Software Technology **48** (2006), no. 2, 69 – 89.

[GBGo9] Thomas Grill, Bettina Biel, and Volker Gruhn, *A pattern approach to mobile interaction design*, it - Information Technology **51** (2009), no. 2, 93–101.

[GJBo5] Elspeth Golden, Bonnie E. John, and Len Bass, *The value of a usability-supporting architectural pattern in software architecture design: a controlled experiment*, ICSE '05: Proc. of the 27th international conference on Software engineering (New York, NY, USA), ACM, 2005, pp. 460–469.

[GPo2] Jonathan Grudin and John Pruitt, *Personas, participatory design and product development: An infrastructure for engagement.*, Proc. of the PDA 2002 (Malmö), 2002, pp. 144–161.

[Grio9] Thomas Grill, *Designing interactions in mixed interaction environments*, Ph.D. thesis, Johannes Kepler University Linz, 2009.

[GToo] Volker Gruhn und Andreas Thiel, *Komponentenmodelle*, Addison-Wesley, 2000.

[Gut94] Thomas A. Gutzwiller, *Das CC RIM-Referenzmodell für den Entwurf von betrieblichen, transaktionsorientierten Informationssystemen*, Physica, Heidelberg, 1994.

[Haio7] Jukka Haikara, *Usability in agile software development: Extending the interaction design process with personas approach*, Agile Processes in Software Engineering and Extreme Programming (2007), 153–156.

[HHGo8] Karin A. Hummel, Andrea Hess, and Thomas Grill, *Environmental context sensing for usability evaluation in mobile HCI by means of small wireless sensor networks*, Proc. of the Mobile Multimedia (MoMM 2008), ACM, 2008, pp. 302–306.

[HHPoo] Derek Hatley Hruschka, Peter Hruschka, and Imtiaz Pirbhai, *Process for system architecture and requirements engineering*, Dorset House Publishing Co Inc., U.S., 2000.

[HNS99] Christine Hofmeister, Robert Nord, and Dilip Soni, *Applied software architecture*, Addison-Wesley Longman, Amsterdam, 1999.

[HSo9] Peter Hruschka und Gernot Starke, *arc42 – Ressourcen für Software-Architekten*, http://www.arc42.de/ (2002–2009), online.

[HTKRo5] Hagen Höpfner, Can Türker und Birgitta König-Ries, *Mobile Datenbanken und Informationssysteme — Konzepte und Techniken*, dpunkt.verlag, Heidelberg, Germany, July 2005.

[IEE90] IEEE, *IEEE Std 610.12-1990 IEEE Standard Glossary of Software Engineering Terminology*, IEEE, http://standards.ieee.org/, 1990.

[Int99] International Organization for Standardization, *Ergonomische Anforderungen für Bürotätigkeiten mit Bildschirmgeräten - Teil 11: Anforderungen an die Gebrauchstauglichkeit; Leitsätze (ISO 9241-11:1998); Deutsche Fassung EN ISO 9241-11:1998*, International Organization for Standardization, 1999.

[Intoo] ———, *Iso 9126-1 (2000): Software engineering - product quality - part 1: Quality model*, International Organization for Standardization, 2000.

[Jac01] Michael A. Jackson, *Problem frames. Analyzing and structuring software development problems*, Addison-Wesley, Harlow, GB, 2001.

[JBR99] Ivar Jacobson, Grady Booch, and James Rumbaugh, *The unified software development process*, Addison-Wesley, 1999.

[JM06] Matt Jones and Gary Marsden, *Mobile interaction design*, John Wiley & Sons, February 2006.

[JMSS07] Natalia Juristo, Ana M. Moreno, and Maria-Isabel Sanchez-Segura, *Analysing the impact of usability on software design*, Journal of Systems and Software **80** (2007), no. 9, 1506–1516.

[KABC96] Rick Kazman, Gregory Abowd, Len Bass, and Paul Clements, *Scenario-based analysis of software architecture*, IEEE Software (1996), no. November, 47–55.

[Kaz96] Rick Kazman, *Tool support for architecture analysis and design*, Proc. of the 2nd International Software Architecture Workshop, 1996.

[KK08] Philip Kotler und Kevin Keller, *Marketing Management*, Prentice Hall, 2008.

[KKB+98] Rick Kazman, Mark Klein, Mario R. Barbacci, T. Longstaff, H. Lipson, and J. Carriere, *The architecture tradeoff analysis method*, Proc. of the IEEE ICECCS, 1998.

[KKLN05] Rick Kazman, Mark Klein, Tony Lattanze, and Linda Northrop, *A basis for analyzing software architecture analysis methods*, Software Quality Journal **13** (2005), 329–355.

[Kor07] Christoph Korzenek, *Usability-Anforderungen für mobile Anwendungen*, Tech. report, Universität Leipzig, 2007.

[Kru98] Philippe Kruchten, *The rational unified process. An introduction*, Addison-Wesley Longman, Amsterdam, 1998.

[Kru04] ———, *An ontology of architectural design decisions.*, 2nd Groningen Workshop on Software Variability Management, 2004.

[LBKK97] Chung-Horng Lung, Sonia Bot, K. Kalaichelvan, and Rick Kazman, *An approach to software architecture analysis for evolution and reusability"*, Proc. of the CASCON, 1997.

[Lit10] Little Springs Design, *Design for mobile. Resources for designing and building mobile apps and sites*, new http://4ourth.com/wiki – http://patterns.littlespringsdesign.com (via http://wayback.archive.org/web/) (letzter Zugriff am 17. Februar 2010), online.

[LRV99] Nico Lassing, D. Rijsenbrij, and Hans van Vliet, *On software architecture analysis of flexibility, complexity of changes: Size isn't everything*, Proc. of the 2nd Nordic Software Architecture Workshop, 1999.

[MD12] Gerard Meszaros and Jim Doble, *A pattern language for pattern writing*, http://www.hillside.net/index.php/a-pattern-language-for-pattern-writing (letzter Zugriff am 12. April 2012), online.

[MM03] Nikunj R. Mehta and Nenad Medvidovic, *Composing architectural styles from architectural primitives*, Proc. of the 9th European Software Engineering Conference and the 10th ACM Sigsoft International Symposium on Foundations of Software Engineering, ACM Press, 2003, pp. 347–350.

[Mol99] Georg Molter, *Integrating saam in domain-centric and reuse- based development processe*, Proc. of the 2nd Nordic Workshop on Software Architecture, 1999.

[MRW77] Jim A. McCall, Paul K. Richards, and Gene F. Walters, *Factors in software quality*, vol. Vol. 1-3, National Technical Information Service, 1977.

[MT00] Nenad Medvidovic and Richard N. Taylor, *A classification and comparison framework for software architecture description languages*, IEEE Trans. Softw. Eng. **26** (2000), no. 1, 70–93.

[Nie93] Jakob Nielsen, *Usability engineering*, Academic Press Limited, 1993.

[Nor04] Donald A. Norman, *Emotional design: Why we love (or hate) everyday things*, Basic Books New York, 2004.

[Nus01] Bashar Nuseibeh, *Weaving together requirements and architectures*, Computer **34** (2001), no. 3, 115–117.

[Obj11] Object Management Group OMG, *The unified modeling language*, http://www.omg.org/spec/UML/ (letzter Zugriff am 8. Februar 2011), online.

[Oul05] Antti Oulasvirta, *The fragmentation of attention in mobile interaction, and what to do with it*, interactions **12** (2005), no. 6, 16–18.

[Pan99] Raj Pandya, *Mobile and personal communication systems and services*, Wiley Publishing, Inc.: Indianapolis, 1999.

[PBG04] Torsten Posch, Klaus Birken, and Michael Gerdom, *Basiswissen Softwarearchitektur*, dpunkt.verlag, 2004.

[PG03] John Pruitt and Jonathan Grudin, *Personas: practice and theory*, DUX '03: Proc. of the 2003 conference on Designing for user experiences (New York, NY, USA), ACM Press, 2003, pp. 1–15.

[Poh06] Klaus Pohl, *Requirements Engineering: Grundlagen, Prinzipien, Techniken*, dpunkt.Verlag GmbH, 2006.

[PRS07] Jenny Preece, Yvonne Rogers, and Helen Sharp, *Interaction design: Beyond human-computer interaction*, 2nd ed., Addison-Wesley, 2007.

[PS15] Anil Patidar and Ugrasen Suman, *A survey on software architecture evaluation methods*, Proc. of the 2nd International Conference on Computing for Sustainable Global Development (INDIACom), 2015.

[RM02] Louis Rosenfeld and Peter Morville, *Information architecture for the world wide web, 2nd edition*, O'Reilly, 2002.

[RPM00] Gruia-Catalin Roman, Gian Pietro Picco, and Amy L.L. Murphy, *Software engineering for mobility: A roadmap.*, Proc. of the Conference on the Future of Software Engineering (ACM Press, ed.), 2000, pp. 241–258.

[RRD06] Tamer Rafla, Pierre N. Robillard, and Michel Desmarais, *Investigating the impact of usability on software architecture through scenarios: A case study on web systems*, J. Syst. Softw. **79** (2006), 415–426.

[RRD07] _____, *A method to elicit architecturally sensitive usability requirements. Its integration into a software development process.*, Software Quality Journal **15** (2007), 117–133.

[SAR02] SARA Working Group, *Software architecture review and assessment (SARA) report*, Tech. report, SARA Working Group, 1999-2002.

[SBG99] Albrecht Schmidt, Michael Beigl, and Hans-Werner Gellersen, *There is more to context than location*, Computers and Graphics **23** (1999), no. 6, 893–901.

[SBJG09] Pia Stoll, Len Bass, Bonnie E. John, and Elspeth Golden, *Supporting usability in product line architectures*, Software Product Lines Conference, SPLC 2009, San Francisco, USA, 2009.

[SC97] Mary Shaw and Paul C. Clements, *A field guide to boxology: Preliminary classification of architectural styles for software systems*, Proc. of the 21st International Computer Software and Applications Conference (IEEE, ed.), 1997, pp. 6–13.

[Sei09] Stefan Seitz, *Pattern-basierte Usability-Szenarien (Diplomarbeit)*, Ludwig-Maximilians-Universität München (2009).

[Sei10] Michael Seiltz, *Mobile-ui.org: Eine web-basierte kollaborative Patternsammlung (Bachelorarbeit)*, Universität Leipzig (2010).

[SEI12] SEI Website, Carnegie Mellon, *Community software architecture definitions*, www.sei.cmu.edu/architecture/start/glossary/community.cfm (letzter Zugriff am 8. Juli 2012), online.

[SG96] Mary Shaw and David Garlan, *Software architecture: Perspectives on an emerging discipline*, Addison-Wesley, 1996.

[SHKB05] Albrecht Schmidt, Paul Holleis, Matthias Kranz und Andreas Butz, *Neue Formen der Interaktion mit Mobilen Geräten*, Tech. Report LMU-MI-2005-2, University of Munich, Dep. of Computer Science - Media Informatics Group, Munich, 2005.

[SN07] Martin Schmettow and Sabine Niebuhr, *A pattern-based usability inspection method. First empirical performance measures and future issues*, Proc. of the 21st BCS HCI Group Annual Conference (HCI) Lancaster, UK (British Computer Society, ed.), vol. 2, Sept. 2007, pp. 99–102.

[Som01] Ian Sommerville, *Software engineering*, 6. ed., Pearson Studium, 2001.

[SR91] Brian Shackel and Simon Richards, *Human factors for informatics usability*, ch. 2 - Context, Framework, Definition, Design and Evaluation, pp. 21–37, Cambridge University Press, Cambridge, 1991.

[Sta05] Gernot Starke, *Effektive Softwarearchitekturen*, 2. Auflage ed., Carl Hanser Verlag München Wien, 2005.

[Sto05] Harald Stoerrle, *UML 2 erfolgreich einsetzen*, Programmer, Addison-Wesley, 2005.

[SW07] Markus Specker and Ina Wentzlaff, *Exploring usability needs by human-computer interaction patterns*, Proc. of the 6th International Workshop on TAsk MOdels and DIAgrams (TAMODIA). Toulouse, France (Springer, ed.), 2007, pp. 254–260.

[Taro3] Peter Tarasewich, *Designing mobile commerce applications*, Communications of the ACM **46** (2003), no. 12, 57–60.

[TGHW06] Bettina Thurnher, Thomas Grill, Karin Hummel, and Roman Weigl, *Exploiting context-awareness for usability evaluation in mobile HCI*, Proc. of the Usability Day IV (Guido Kempter and Philipp von Hellberg, eds.), Pabst Science Publisher, June 2006, pp. 109–113.

[Tido5] Jenifer Tidwell, *Designing interfaces*, O'Reilly, 2005.

[TOTK04] Sakari Tamminen, Antti Oulasvirta, Kalle Toiskallio, and Anu Kankainen, *Understanding mobile contexts*, Personal Ubiquitous Comput. **8** (2004), no. 2, 135–143.

[VFCS15] Jéssyka Vilela, Bruno Figueiredo, Jaelson Castro, and Monique Soares, *Usability and software architecture: a literature review*, Proc. of the IXth Brazilian Symposium on Components, Architectures and Reuse Software, 2015.

[VKZ04] Markus Völker, Michael Kircher, and Uwe Zdun, *Remoting patterns*, John Wiley & Sons, 2004.

[VV02] Upkar Varshney and Ron Vetter, *Mobile commerce: Framework, applications and networking support.*, Mobile Networks and Applications **7** (2002), 185–198.

[vW12] Martijn van Welie, *A pattern library for interaction design, (last visited 4 apr 2011)*, http://www.welie.com (letzter Zugriff am 10. April 2012), online.

[WF11] Tim Wellhausen and Andreas Fiesser, *How to write a pattern? A guideline for first-time pattern authors*, Proc. of the 16th European Conference on Pattern Languages of Programs (EuroPLoP 2011), 2011.

[WKRSS09] Matthias Wermke, Kathrin Kunkel-Razum und Werner Schulze-Stubenrecht, *Duden Rechtschreibung der deutschen Sprache*, 21. ed., Dudenverlag, 2009.

[ZBJ04] Liming Zhu, Muhammad Ali Babar, and Ross Jeffery, *Mining patterns to support software architecture evaluation*, Proc. of the Fourth Working IEEE/IFIP Conference on Software Architecture (WICSA' 04), SA, Scenarios, Patterns 2004.

[ZCTT05] Ping Zhang, Jane Carey, Dov Te'eni, and Marilyn Tremaine, *Integrating human-computer-interaction development into the systems development life cycle: A methodology*, Communications of the ACM **15** (2005), 512–543.

DANKE

Vielen herzlichen Dank für die wissenschaftliche, berufliche und private Unterstützung dieser Forschung.

SELBSTÄNDIGKEITSERKLÄRUNG

Den vorliegenden Text habe ich selbständig und ohne unzulässige fremde Hilfe verfasst. Aussagen und Erkenntnisse von anderen habe ich nach bestem Wissen und Gewissen referenziert.

Leipzig, 08.07.2016

Bettina Biel

KOLOPHON

Erstellt wurde dieses Dokument mit der LaTeX -Vorlage `classicthesis` von André Miede, inspiriert von Robert Bringhurst's *„The Elements of Typographic Style"*. Das `classicthesis`-Paket ist für alle Autoren kostenfrei erhältlich via `http://code.google.com/p/classicthesis/` (für LaTeX und LyX). André Miede wünscht sich dafür nur eine Postkarte. `http://postcards.miede.de/`. Danke!